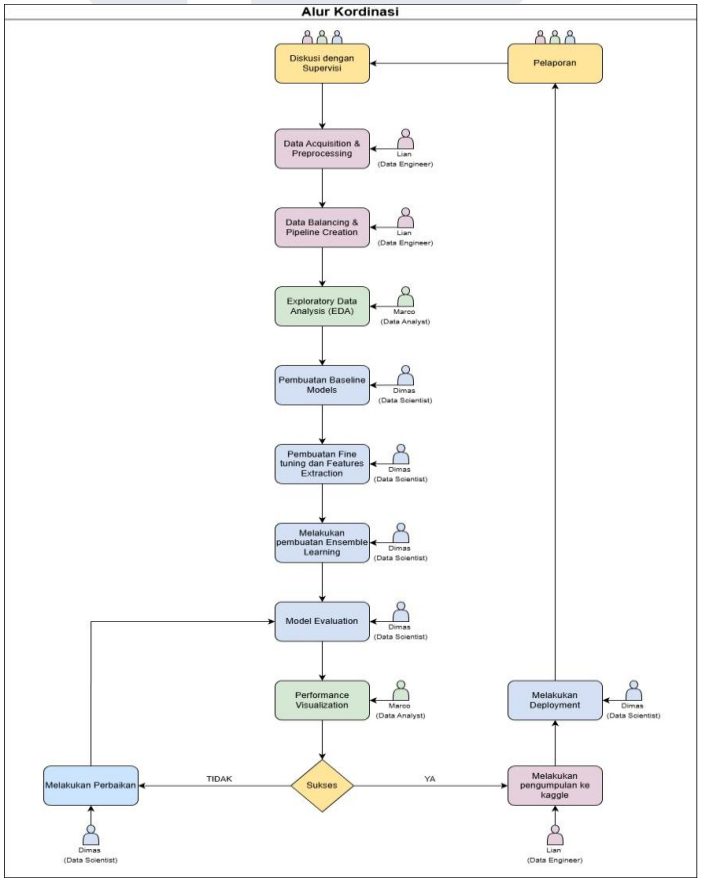
BAB III

PELAKSANAAN PRO-STEP : ROAD TO CHAMPION

3.1 Kedudukan dan Koordinasi

Dalam pelaksanaan PRO-STEP: Road to Champion Program, penulis berpartisipasi sebagai salah satu anggota tim dalam Data Science Competition (DSC) LOGIKA UI 2025, yang merupakan bagian dari rangkaian acara LOGIKA UI (Lomba Karya dan Inovasi Mahasiswa Komputer Universitas Indonesia). Dalam kompetisi ini, penulis menempati posisi sebagai Data Scientist sekaligus penanggung jawab utama dalam pemodelan dan analisis hasil prediksi berbasis deep learning.



Gambar 3.1 Bagan Alur Koordinasi

Gambar 3.1 menunjukkan alur koordinasi lengkap tim Data Science Competition (DSC) LOGIKA UI 2025 dalam mengembangkan sistem klasifikasi rumah adat Nusantara berbasis deep learning. Diagram ini menggambarkan pembagian peran yang jelas antar anggota tim, alur kerja sekuensial dari tahap data acquisition hingga deployment, serta mekanisme iterasi untuk optimalisasi model sesuai dengan batasan waktu kompetisi.

Proses dimulai dengan diskusi bersama Supervisor (Monika Evelin Johan, S.Kom., M.M.S.I.) yang melibatkan seluruh anggota tim. Dalam tahap ini, supervisor memberikan arahan teknis, menetapkan target kompetisi, dan menyepakati pembagian tanggung jawab sesuai keahlian masing-masing anggota. Pembagian peran dalam tim dirancang mengikuti alur kerja data science pipeline standar, di mana Lian Wira Manuel Maharaja sebagai Data Engineer sekaligus Ketua Tim bertanggung jawab pada tahap awal pipeline yaitu Data Acquisition & Preprocessing serta Data Balancing & Pipeline Creation untuk memastikan dataset berkualitas tinggi dan seimbang antar kelas. Sebagai ketua tim, Lian juga berperan dalam koordinasi keseluruhan proyek, monitoring timeline, dan komunikasi utama dengan supervisor.

Setelah data siap, Marco Naftali Stevenson (Data Analyst) melakukan Exploratory Data Analysis (EDA) untuk memahami karakteristik dataset, distribusi kelas, dan pola visual pada citra rumah adat. Hasil EDA ini menjadi dasar dalam pemilihan arsitektur model dan strategi feature extraction yang akan diterapkan pada tahap pemodelan. Dimas Aji Haritson (Data Scientist dan Penulis) kemudian mengembangkan model klasifikasi melalui tiga tahapan utama: Pembuatan Baseline Models menggunakan enam arsitektur CNN populer (VGG16, Xception, DenseNet121, EfficientNetB0, EfficientNetB3, MobileNetV2), Pembuatan Fine-tuning dan Features Extraction dengan mengombinasikan fitur HOG dan DenseNet121, serta Melakukan pembuatan Ensemble Learning dengan variasi learning rate yang mengintegrasikan HOG, DenseNet121, dan

EfficientNetB3.

Setiap model yang dikembangkan melalui tahap Model Evaluation di mana Dimas menghitung berbagai metrics performa seperti Macro F1-score, precision, recall, dan confusion matrix pada validation set. Hasil evaluasi ini kemudian divisualisasikan oleh Marco dalam tahap Performance Visualization yang mencakup confusion matrix heatmap, training curves, dan comparison charts untuk memudahkan analisis performa model. Berdasarkan hasil visualisasi ini, tim melakukan evaluasi bersama untuk menentukan apakah model sudah optimal atau masih memerlukan perbaikan.

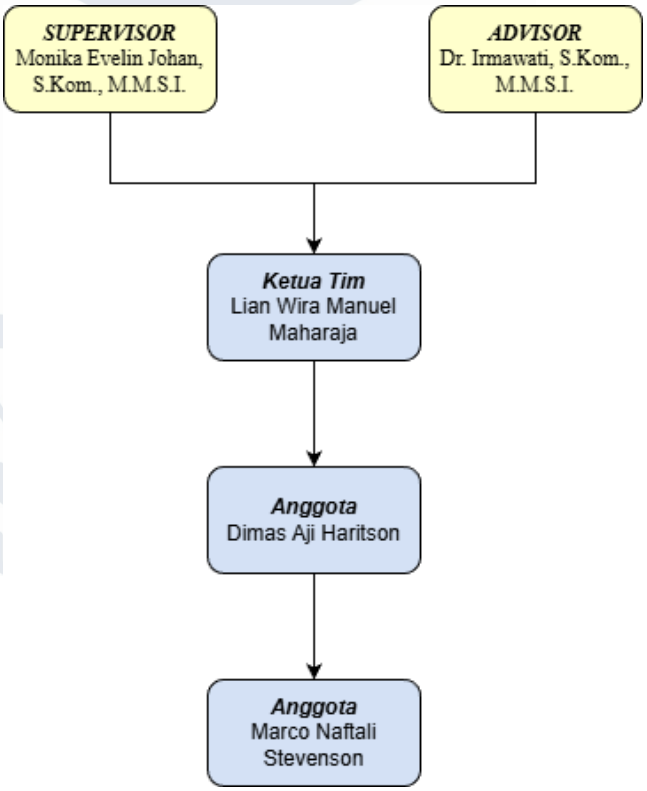
Kriteria keberhasilan model tidak hanya berdasarkan pencapaian target metrics tertentu, tetapi juga mempertimbangkan apakah performa masih dapat ditingkatkan dalam waktu yang tersisa sebelum deadline submission kompetisi. Jika berdasarkan analisis tim model masih memiliki potensi peningkatan yang signifikan dan waktu masih memungkinkan, Dimas melakukan perbaikan model melalui hyperparameter tuning, adjustment arsitektur, atau penambahan teknik augmentasi. Proses iterasi ini berlangsung secara berulang hingga tim memutuskan bahwa model telah mencapai performa optimal yang realistis dalam batasan waktu yang ada, atau ketika deadline submission sudah semakin dekat sehingga tim harus segera melakukan finalisasi model.

Ketika model final telah ditentukan, proses berlanjut ke tahap pengumpulan ke Kaggle di mana Lian sebagai Ketua Tim bertanggung jawab menggenerate predictions pada test set menggunakan ensemble model terbaik, menyiapkan file submission dalam format CSV, dan melakukan upload ke platform Kaggle LOGIKA UI untuk mendapatkan score dari sistem leaderboard. Hasil submission menunjukkan Public Score 0.6648 dan Private Score 0.6058, menempatkan tim pada peringkat 60 dari 94 peserta. Setelah submission selesai, Dimas melakukan Deployment

untuk membuat aplikasi web interaktif menggunakan Streamlit yang di-deploy melalui Ngrok.

Secara struktural, kegiatan ini berada di bawah bimbingan Dr. Irmawati, S.Kom., M.M.S.I. selaku Dosen Pembimbing dari Program Studi Sistem Informasi, Universitas Multimedia Nusantara (UMN). Beliau berperan memberikan arahan akademik, pembimbingan metodologis, serta memastikan bahwa pelaksanaan proyek berjalan sesuai dengan prinsip ilmiah, etika penelitian, dan capaian pembelajaran yang diharapkan.

Selain itu, pelaksanaan proyek juga berada dalam pengawasan Monika Evelin Johan, S.Kom., M.M.S.I. selaku Supervisor, yang memberikan panduan teknis, evaluasi terhadap progres kerja tim, serta memastikan kesesuaian proyek dengan target kompetisi dan standar profesional di bidang data science.



Gambar 3.2 Diagram Kedudukan Organisasi

Koordinasi dilakukan secara berjenjang dan kolaboratif. Dosen pembimbing internal memberikan pengarahan dalam aspek konseptual dan metodologis, sedangkan supervisor berperan dalam memantau pelaksanaan teknis dan capaian hasil. Setiap anggota tim juga berkoordinasi secara rutin melalui rapat internal daring dan grup komunikasi.

1. Struktur Kedudukan dan Peran Anggota Tim

1) Dosen Pembimbing Internal: Dr. Irmawati, S.Kom., M.M.S.I.
   a. Memberikan bimbingan akademik, evaluasi, serta memastikan keaslian dan relevansi proyek dengan bidang ilmu yang ditekuni.
   b. Melakukan supervisi berkala terhadap laporan kemajuan dan hasil akhir proyek.
2) Supervisor Lapangan: Monika Evelin Johan, S.Kom., M.M.S.I.
   a. Memberikan arahan teknis terkait pengembangan model machine learning dan pengelolaan data.
   b. Menilai efisiensi implementasi dan hasil pengujian model pada kompetisi.
3) Tim Kompetisi DSC LOGIKA UI 2025:
   a. Lian Wira Manuel Maharaja – Data Engineer
      Bertanggung jawab atas *data acquisition*, *data preprocessing*, dan pembuatan *data pipeline* yang efisien.
   b. Marco Naftali Stevenson – Data Analyst (Koordinator Tim)
      Melakukan *Exploratory Data Analysis (EDA)*, menyusun visualisasi data, serta mengembangkan *dashboard insight*.
   c. Dimas Aji Haritson – Data Scientist (Penulis)
      Mendesain dan mengembangkan model *deep learning* berbasis *Convolutional Neural Network (CNN)* untuk klasifikasi gambar rumah adat Nusantara.
      Melakukan *hyperparameter tuning* serta membandingkan tiga model berbeda (VGG16, Xception, dan EfficientNetB3) untuk menentukan model dengan akurasi tertinggi.
      Menyusun laporan hasil analisis performa model dan

berkontribusi dalam penyusunan laporan akhir kompetisi.

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

Bagian ini menjelaskan secara detail rangkaian kegiatan yang dilakukan penulis selama mengikuti Pro-Step: Road to Champion Program, yang berfokus pada partisipasi dalam Data Science Competition LOGIKA UI 2025.

Selama prosesnya, penulis berkoordinasi dengan Supervisor (Monika Evelin Johan, S.Kom., M.M.S.I.) yang berperan dalam memberikan bimbingan teknis dan pengarahan proyek, serta Advisor (Dr. Irmawati, S.Kom., M.M.S.I.) yang berperan dalam pembimbingan akademik dan penyusunan laporan kegiatan.

Kegiatan dilakukan secara bertahap mulai dari tahap inisiasi ide lomba, proses analisis dan pengolahan data, hingga tahap bimbingan laporan. Setiap minggu memiliki fokus kegiatan yang berbeda sesuai dengan jadwal dan target yang telah disepakati bersama pembimbing.

Tabel 3.1 Detail Pekerjaan yang Dilakukan PRO-STEP : Road to Champion Program

No	Minggu	Proyek / Tahapan	Keterangan
1	1	Persiapan Lomba LOGIKA UI	Memulai rangkaian lomba LOGIKA UI, menentukan tema Data Science Competition, membagi peran dan tugas dalam kelompok, serta melakukan pendaftaran lomba.
2	2	Eksplorasi Model dan Dataset	Melakukan eksplorasi awal terhadap dataset citra rumah adat, latihan kelompok, pengujian berbagai model klasifikasi citra, penerapan feature engineering awal, serta analisis pola data.
3	3	Pengembangan	Mengembangkan pipeline pemrosesan

No	Minggu	Proyek / Tahapan	Keterangan
		Pipeline Data	data menggunakan ImageDataGenerator, melakukan koordinasi dengan supervisi, serta mengikuti bimbingan kedua terkait progres pengembangan model.
4	4	Eksplorasi Model Lanjutan	Melanjutkan eksplorasi pipeline data untuk pengujian model lanjutan serta mengikuti technical meeting babak penyisihan kompetisi.
5	5–6	Optimasi Model dengan F1-Score	Melakukan eksplorasi dan evaluasi model dengan fokus pada optimasi metrik macro F1-score untuk memperoleh validasi performa yang lebih seimbang antar kelas.
6	7	Pembuatan Base Model	Mengembangkan dan menguji beberapa base model CNN, yaitu VGG16, Xception, MobileNetV2, EfficientNet, dan DenseNet, serta melakukan analisis awal terhadap hasil performa masing-masing model.
7	8	Fine-Tuning Model	Melakukan proses fine tuning pada model Xception, DenseNet, dan EfficientNetB3, dengan capaian performa awal berupa akurasi macro pada kisaran 60–70%.
8	9	Implementasi HOG dan Model	Mempelajari dan mengimplementasikan metode Histogram of Oriented Gradients (HOG), melakukan uji coba klasifikasi citra berbasis HOG, serta mengembangkan model HOG + DenseNet.
9	10	Pengembangan Multi-Architecture	Mengembangkan model dengan menambahkan EfficientNetB3 dan ResNet ke dalam skema , sehingga diperoleh akurasi dan macro average sekitar 85%.

No	Minggu	Proyek / Tahapan	Keterangan
10	11	Evaluasi dan Rekapitulasi Model	Melakukan fine tuning ulang pada metode dengan enam fitur, mengatasi potensi data leakage, melakukan validasi akhir, serta merekap hasil evaluasi base model dan model.

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

Bagian ini berupa penjelasan secara umum mengenai pekerjaan yang dilakukan penulis selama proses *PRO-STEP : Road to Champion Program.*

3.3.1 Proses Pelaksanaan

Pelaksanaan tugas dalam PRO-STEP: Road to Champion Program pada jalur kompetisi DSC LOGIKA UI 2025 dilaksanakan melalui rangkaian proses pemodelan yang terstruktur, berlandaskan pembagian peran dalam tim. Bagian ini berfokus pada aktivitas pemodelan yang mencakup pengembangan, pengujian, evaluasi, serta implementasi model machine learning berbasis deep learning sesuai dengan cakupan tanggung jawab sebagai Data Scientist. Seluruh proses dikerjakan setelah data melalui tahap pembersihan dan eksplorasi awal oleh Data Engineer dan Data Analyst, sehingga pemodelan dapat dimulai dari dataset yang telah siap digunakan.

Tahapan pemodelan yang dilakukan meliputi pengembangan baseline model, penerapan augmentasi dan regularisasi, proses fine-tuning terhadap arsitektur deep learning, optimasi hyperparameter, serta evaluasi performa menggunakan metrik seperti accuracy, precision, recall, dan F1-score. Evaluasi model difokuskan pada pemerataan performa antar kelas, dengan

penggunaan Macro F1-score sebagai metrik utama untuk menilai kestabilan prediksi pada kondisi distribusi data yang tidak seimbang.

Selain tahap pengembangan dan evaluasi model, proses pelaksanaan juga mencakup tahap implementasi dan deployment sistem klasifikasi citra rumah adat. Model terbaik hasil pendekatan dan ensemble diintegrasikan ke dalam aplikasi berbasis web menggunakan Streamlit, kemudian dipublikasikan secara daring melalui layanan Ngrok. Tahap ini bertujuan untuk memvalidasi kesiapan model dalam skenario penggunaan nyata serta memastikan bahwa sistem yang dikembangkan tidak hanya berfungsi secara eksperimental, tetapi juga dapat digunakan secara interaktif oleh pengguna akhir.

Seluruh proses didokumentasikan secara sistematis dalam bentuk kode program, grafik hasil pelatihan, laporan evaluasi model, serta dokumentasi aplikasi deployment. Dokumentasi tersebut berfungsi sebagai bukti pelaksanaan proses pengembangan model yang mengikuti alur kerja data science secara end-to-end dan sesuai dengan standar kompetisi DSC LOGIKA UI 2025.

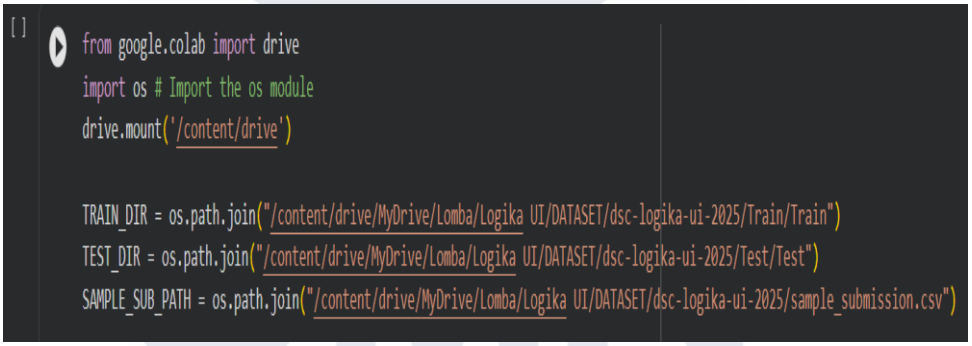
3.3.1.1 Pengembangan Baseline Model

1. **Persiapan environment dan verifikasi dataset**

```
!pip install tensorflow keras numpy pandas scikit-learn matplotlib
Show hidden output
import tensorflow as tf
from tensorflow.keras.applications import Xception
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Gambar 3.3 Persiapan Environment

Gambar 3.3 menunjukkan tahap awal proses pengembangan model, yaitu instalasi dependensi dan import library yang diperlukan dalam pembangunan sistem klasifikasi berbasis deep learning. Pada bagian ini, seluruh paket utama seperti TensorFlow, Keras, NumPy, Pandas, Scikit-learn, dan Matplotlib diinstal untuk memastikan lingkungan kerja siap digunakan. Setelah proses instalasi, dilakukan import modul–modul spesifik yang akan digunakan pada tahap pemodelan, seperti arsitektur Xception untuk transfer learning, ImageDataGenerator untuk preprocessing dan augmentasi gambar, serta layer tambahan seperti GlobalAveragePooling2D dan Dense sebagai blok klasifikasi model. Library lain seperti Adam digunakan sebagai optimizer, sementara Scikit-learn menyediakan fungsi evaluasi F1-score dan utilitas pembagian dataset. Selain itu, NumPy, Pandas, dan Matplotlib membantu dalam pengolahan data, pembuatan dataframe, dan visualisasi proses pelatihan.

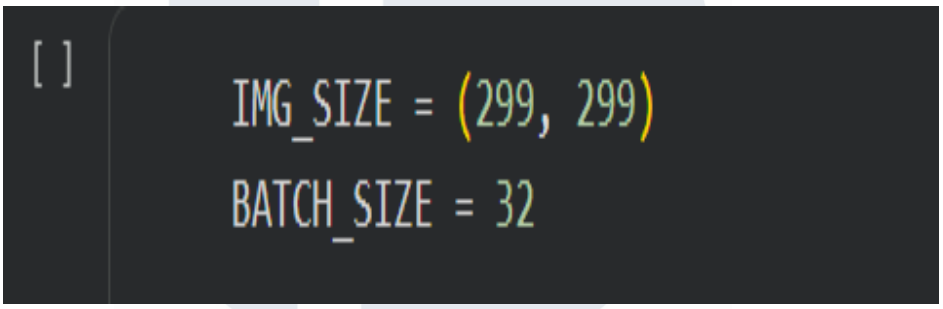


Gambar 3.4 Pemanggilan Dataset

Gambar 3.4 memperlihatkan proses inisialisasi dan pengaturan direktori data yang akan digunakan dalam pengembangan model. Pada tahap ini, Google Drive di-mount ke lingkungan kerja Google Colab menggunakan perintah drive.mount('/content/drive'), sehingga seluruh file dataset dapat diakses secara langsung oleh notebook. Setelah proses mounting berhasil, dilakukan import modul os untuk mempermudah manipulasi path file secara dinamis. Selanjutnya, ditentukan tiga direktori utama, yaitu TRAIN_DIR, TEST_DIR, dan SAMPLE_SUB_PATH, masing-masing menunjuk pada lokasi dataset training, dataset testing, serta

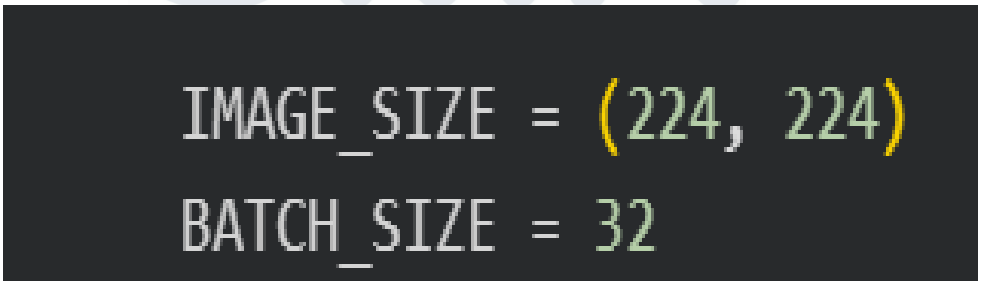
file contoh submission yang disediakan oleh penyelenggara kompetisi. Pendefinisian path ini penting dilakukan di awal agar seluruh proses preprocessing, data loading, dan generasi prediksi dapat berlangsung secara konsisten menggunakan struktur folder yang terorganisir. Tahap ini sekaligus memastikan bahwa environment telah terhubung dengan sumber data yang benar sebelum masuk ke tahap persiapan data dan pembuatan generator citra.

2. Pembuatan Hyperparamter Model



Gambar 3.5 Hyperparamter model Xception

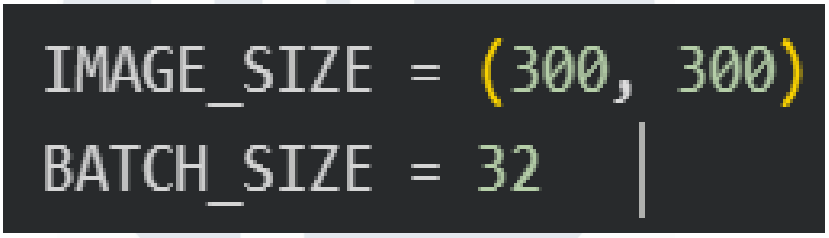
Gambar 3.5 menunjukkan tahapan ukuran citra dan batch size ditentukan terlebih dahulu, yaitu IMAGE_SIZE = (299, 299) yang disesuaikan secara spesifik dengan arsitektur Xception, serta BATCH_SIZE = 32 untuk menjaga keseimbangan antara stabilitas pembelajaran dan efisiensi komputasi.



Gambar 3.6 Hyperparamter model VGG16,
DenseNet121, MobileNetV2, EfficientNetB0

Perbedaan hyperparameter, khususnya ukuran input citra, antara Xception, VGG16, EfficientNet, dan arsitektur CNN lainnya bukanlah kebetulan, melainkan merupakan konsekuensi langsung dari desain internal

masing-masing model. Xception menggunakan ukuran input 299×299 karena arsitektur depthwise separable convolution miliknya dioptimalkan oleh Google pada resolusi tersebut, agar mampu menangkap detail spasial yang lebih halus tanpa menambah kompleksitas secara signifikan. Sebaliknya, VGG16 menggunakan ukuran standar 224×224, mengikuti tradisi arsitektur CNN klasik yang diperkenalkan pada ImageNet 2014. Model ini memprioritaskan struktur bertingkat yang sederhana dan konsisten, sehingga ukuran 224×224 dinilai paling stabil untuk konvolusi berlapis-lapis yang dalam.



Gambar 3.7 Hyperparamter model EfficientNetB3

EfficientNet memiliki ukuran input yang berbeda untuk setiap variannya, misalnya EfficientNet-B0 menggunakan 224×224, sedangkan EfficientNet-B3 menggunakan 300×300. Hal ini terjadi karena EfficientNet mengadopsi pendekatan *compound scaling*, yaitu metode yang secara sistematis menambah resolusi, jumlah layer, dan jumlah filter secara bersamaan untuk mencapai akurasi optimal. Dengan demikian, ukuran seperti 300×300 merupakan hasil perhitungan scaling yang dirancang agar model tetap efisien secara komputasi meskipun resolusinya meningkat. Oleh karena itu, perbedaan ukuran input, batch size, atau epoch antar model merupakan bagian dari karakteristik desain masing-masing arsitektur yang ditetapkan berdasarkan eksperimen asli peneliti pembuat model. Perbedaan ini perlu dipertahankan agar setiap model dapat bekerja secara optimal sesuai rancangan awalnya.

3. Pengembangan Arsitektur Model Baseline

```
base_vgg = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))
base_vgg.trainable = False  # freeze backbone

x = base_vgg.output
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)
outputs = Dense(num_classes, activation='softmax')(x)

model_vgg = Model(inputs=base_vgg.input, outputs=outputs)
model_vgg.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy', metrics=['accuracy'])
model_vgg.summary()
```

Gambar 3.8 Pengembangan Arsitektur Model Baseline

Pada tahap pengembangan model baseline, penulis membangun beberapa arsitektur transfer learning sebagai pembanding dari model utama. Salah satu contoh implementasi baseline tersebut adalah penggunaan model VGG16, yang menunjukkan pola umum pembangunan model pada proses ini. Setiap arsitektur baseline dibangun dengan struktur yang hampir sama, yaitu: memuat pretrained backbone tanpa top layer, membekukan bobot (freezing), menambahkan custom classification head, dan mengompilasi model untuk tahap pelatihan awal. Perbedaan utama antar model baseline umumnya hanya terletak pada ukuran input standar masing-masing arsitektur, seperti 224×224 untuk VGG16, 299×299 untuk Xception, dan 300×300 untuk EfficientNet [9].

Pada contoh baseline VGG16, model dimuat menggunakan parameter include_top=False sehingga lapisan fully-connected bawaan ImageNet dihilangkan. Backbone kemudian dibekukan (trainable=False) agar fungsi pretrained sebagai feature extractor tetap terjaga pada tahap awal. Selanjutnya, arsitektur classifier dibentuk dengan meratakan output fitur menggunakan Flatten(), diikuti oleh Dense berjumlah 256 neuron dengan aktivasi ReLU untuk mempelajari representasi non-linear tambahan. Mekanisme regularisasi ditambahkan melalui Dropout(0.3) guna mengurangi risiko overfitting pada dataset kompetisi. Lapisan akhir menggunakan Dense sejumlah num_classes dengan aktivasi softmax sebagai output probabilistik untuk klasifikasi multi-kelas.

Setelah struktur model selesai, proses kompilasi dilakukan menggunakan optimizer Adam dengan learning rate kecil yaitu 1e-4. Pemilihan LR ini disesuaikan dengan karakteristik VGG16 yang memiliki jumlah parameter cukup besar dan lebih sensitif terhadap pembaruan bobot dibandingkan arsitektur yang lebih modern seperti Xception atau EfficientNet. Meskipun demikian, pola pembangunan dan proses kompilasi secara umum tetap sama untuk seluruh model baseline [10]. Perbedaan konfigurasi hanya terletak pada aspek yang terkait dengan arsitektur masing-masing, terutama ukuran input default dan kompleksitas layer internal. Dengan pendekatan ini, setiap model baseline memiliki struktur yang seragam namun tetap optimal mengikuti karakteristik arsitekturnya.

4. Konfigurasi Pelatihan dan Eksekusi Training Model Baseline

```
es = EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)
rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2)

history_vgg = model_vgg.fit(
    train_gen,
    validation_data=val_gen,
    epochs=EPOCHS,
    callbacks=[es, rlr]
)
```

Gambar 3.9 Konfigurasi Pelatihan dan Eksekusi Training Model Baseline

Pada langkah ini, proses beralih dari tahap perancangan arsitektur model baseline menuju tahap konfigurasi serta eksekusi pelatihan model. Agar proses training berlangsung secara stabil dan efisien, digunakan dua callback utama, yaitu EarlyStopping dan ReduceLROnPlateau. EarlyStopping berfungsi untuk menghentikan pelatihan secara otomatis apabila nilai validation loss tidak menunjukkan perbaikan setelah sejumlah patience yang ditentukan (empat epoch). Mekanisme ini bertujuan untuk mencegah terjadinya overfitting serta memastikan bahwa model tidak terus dilatih pada kondisi ketika performanya justru menurun. Selain itu, parameter restore_best_weights=True memastikan bobot terbaik selama

proses pelatihan dipulihkan saat training selesai.

Sementara itu, ReduceLROnPlateau berfungsi menurunkan learning rate sebesar faktor 0.3 apabila validation loss stagnan selama dua epoch. Strategi adaptif ini memungkinkan model untuk melakukan pembelajaran yang lebih halus pada fase-fase akhir pelatihan, khususnya ketika gradien sudah mengecil dan model membutuhkan langkah optimasi yang lebih konservatif.

Setelah callback dikonfigurasikan, model baseline kemudian dilatih menggunakan metode fit(), dengan train_gen sebagai data pelatihan dan val_gen sebagai data validasi. Proses pelatihan dijalankan selama jumlah epoch yang telah ditentukan pada model baseline (misalnya 10 epoch untuk VGG16 dan beberapa arsitektur lain), serta dengan pengawasan langsung terhadap perkembangan validation loss. Dengan demikian, langkah ini menjadi bagian penting dalam memastikan bahwa model baseline tidak hanya berjalan, tetapi juga belajar secara optimal melalui mekanisme kontrol otomatis yang menjaga stabilitas, efisiensi, serta kualitas hasil akhir model.

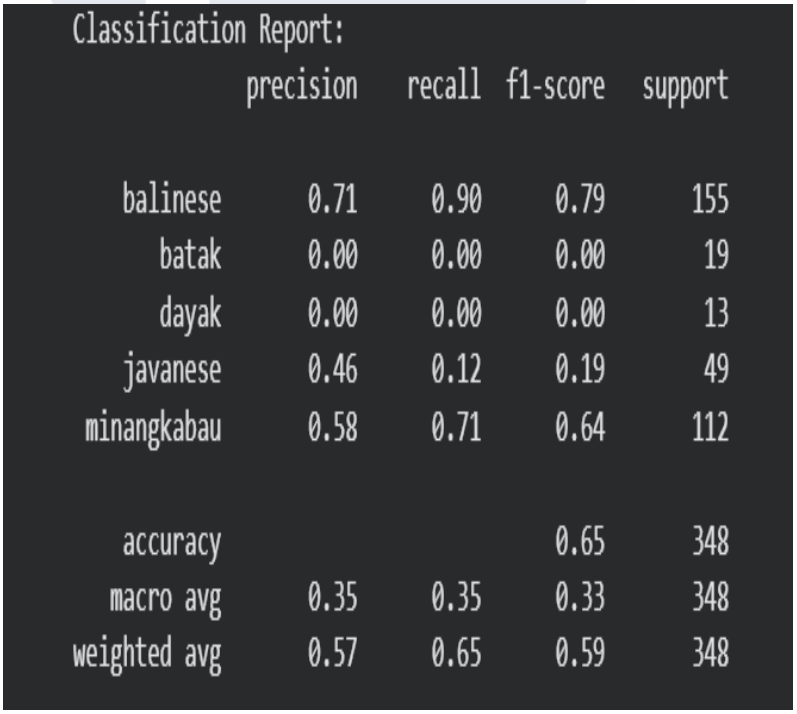
5. **Evaluasi Model Baseline**

```
# predict on val
val_gen.reset()
pred = model_vgg.predict(val_gen, verbose=1)
pred_labels = np.argmax(pred, axis=1)
true_labels = val_gen.classes
target_names = list(train_gen.class_indices.keys())

# classification report
print("Classification Report (VGG16 - Baseline):")
print(classification_report(true_labels, pred_labels, target_names=target_names, digits=4))

macro_f1_vgg = f1_score(true_labels, pred_labels, average='macro')
print("Macro F1 (VGG16 baseline):", macro_f1_vgg)
```

Gambar 3.10 Evaluasi Model Baseline

Pada Gambar 3.10, model yang telah selesai dilatih kemudian dievaluasi menggunakan data validasi untuk mengukur performanya secara

objektif. Proses evaluasi dimulai dengan melakukan prediksi terhadap seluruh data validasi menggunakan model baseline yang telah dibangun. Hasil prediksi yang berupa probabilitas setiap kelas kemudian dikonversi menjadi label kelas melalui fungsi argmax. Label tersebut selanjutnya dibandingkan dengan label sebenarnya dari data validasi. Dari perbandingan ini dihasilkan *classification report* yang berisi metrik penting seperti precision, recall, dan F1-score untuk masing-masing kelas. Selain itu, dihitung juga macro F1-score, yaitu metrik utama yang digunakan dalam permasalahan ini karena mampu menilai performa model secara seimbang pada setiap kelas tanpa dipengaruhi distribusi data yang tidak merata. Nilai macro F1 inilah yang digunakan untuk menilai kualitas model baseline dan menjadi acuan pembanding bagi model-model berikutnya.



Classification Report:

	precision	recall	f1-score	support
balinese	0.71	0.90	0.79	155
batak	0.00	0.00	0.00	19
dayak	0.00	0.00	0.00	13
javanese	0.46	0.12	0.19	49
minangkabau	0.58	0.71	0.64	112
accuracy			0.65	348
macro avg	0.35	0.35	0.33	348
weighted avg	0.57	0.65	0.59	348

Gambar 3.11 Classification Report Model Xception Baseline

Gambar 3.11 menunjukkan hasil evaluasi baseline dari model Xception yang digunakan sebagai salah satu arsitektur deep learning untuk klasifikasi citra etnis. Pada hasil baseline ini, Xception memperoleh akurasi

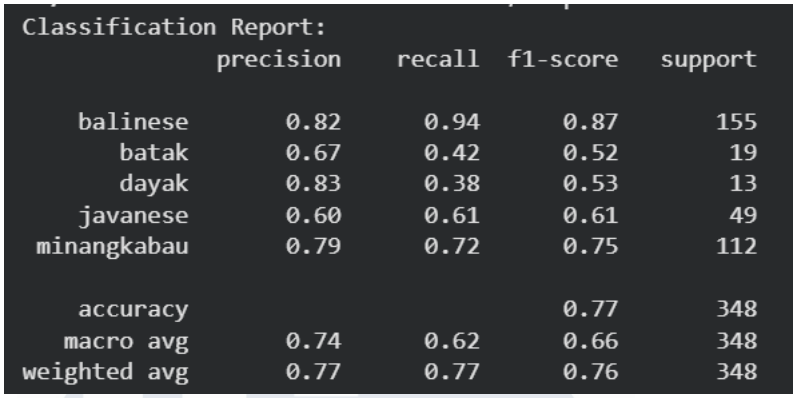
sebesar 65% dengan nilai macro F1 sebesar 0.33. Nilai performa ini tergolong rendah terutama karena beberapa kelas seperti batak dan dayak tidak berhasil diprediksi sama sekali (precision dan recall = 0). Hal ini mengindikasikan bahwa Xception, pada tahap baseline tanpa fine-tuning, belum mampu mengekstraksi pola visual yang cukup representatif dari seluruh kelas, terutama kelas minoritas dengan jumlah data kecil. Meskipun demikian, performanya pada kelas balinese dan minangkabau masih relatif baik, yang menunjukkan bahwa model lebih sensitif terhadap kelas dengan distribusi data besar.

Classification Report (VGG16 - Baseline):

	precision	recall	f1-score	support
balinese	0.7633	0.8323	0.7963	155
batak	0.4500	0.4737	0.4615	19
dayak	0.5556	0.3846	0.4545	13
javanese	0.6486	0.4898	0.5581	49
minangkabau	0.6991	0.7054	0.7022	112
accuracy			0.7069	348
macro avg	0.6233	0.5771	0.5945	348
weighted avg	0.7016	0.7069	0.7014	348

Gambar 3.12 Classification Report Model VGG16 Baseline

Gambar 3.12 menyajikan hasil baseline model VGG16 yang menunjukkan peningkatan signifikan dibandingkan Xception. Model ini mencatat akurasi sebesar 70.69% dan macro F1 sebesar 0.5945. VGG16 berhasil memberikan prediksi yang jauh lebih stabil pada seluruh kelas, termasuk kelas minoritas seperti batak dan dayak. Meskipun performa kedua kelas ini masih lebih rendah dibandingkan kelas dominan, model menunjukkan adanya kemampuan generalisasi yang lebih baik. Peningkatan performa ini didukung oleh arsitektur VGG16 yang lebih sederhana dan cenderung stabil pada dataset berukuran kecil–menengah. Hal ini menunjukkan bahwa VGG16 menjadi baseline yang lebih kuat dibandingkan arsitektur sebelumnya.

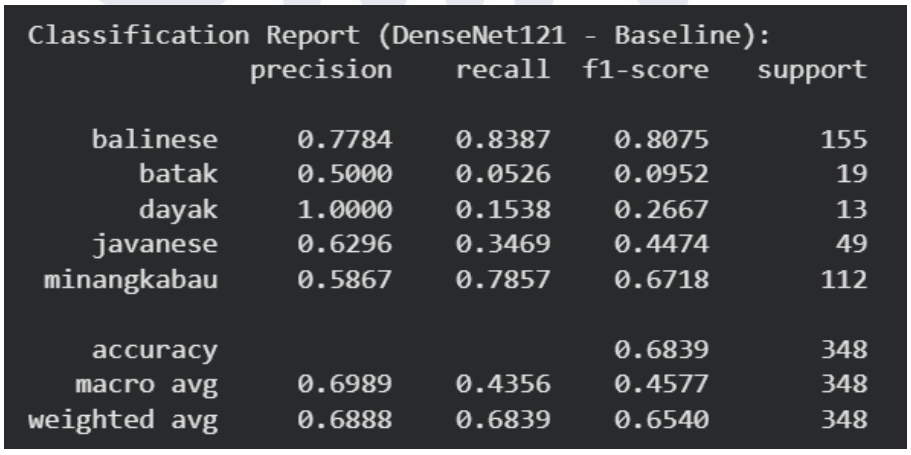


Classification Report:

	precision	recall	f1-score	support
balinese	0.82	0.94	0.87	155
batak	0.67	0.42	0.52	19
dayak	0.83	0.38	0.53	13
javanese	0.60	0.61	0.61	49
minangkabau	0.79	0.72	0.75	112
accuracy			0.77	348
macro avg	0.74	0.62	0.66	348
weighted avg	0.77	0.77	0.76	348

Gambar 3.13 Classification Report Model EfficientNetB3 Baseline

Pada Gambar 3.13, hasil baseline EfficientNetB3 memperlihatkan performa tertinggi di antara seluruh model yang diuji pada tahap baseline, dengan akurasi mencapai 77% dan macro F1 sekitar 0.66. Model ini mampu memberikan prediksi yang konsisten dengan nilai precision dan recall yang relatif tinggi pada hampir semua kelas, khususnya balinese, minangkabau, dan javanese. Performa tinggi EfficientNetB3 disebabkan oleh mekanisme compound scaling yang menyeimbangkan kedalaman, lebar, dan resolusi jaringan sehingga model lebih efisien dalam mengekstraksi fitur visual. Bahkan pada kelas minoritas seperti batak dan dayak, EfficientNetB3 menunjukkan peningkatan performa yang cukup baik dibandingkan model sebelumnya.



Classification Report (DenseNet121 - Baseline):

	precision	recall	f1-score	support
balinese	0.7784	0.8387	0.8075	155
batak	0.5000	0.0526	0.0952	19
dayak	1.0000	0.1538	0.2667	13
javanese	0.6296	0.3469	0.4474	49
minangkabau	0.5867	0.7857	0.6718	112
accuracy			0.6839	348
macro avg	0.6989	0.4356	0.4577	348
weighted avg	0.6888	0.6839	0.6540	348

Gambar 3.14 Classification Report Model DenseNet121 Baseline

Gambar 3.14 menampilkan hasil baseline DenseNet121 dengan akurasi 68.39% dan macro F1 sebesar 0.4577. Meskipun performanya berada di atas Xception, hasil DenseNet121 masih kurang stabil bila dibandingkan VGG16 dan EfficientNet. Model menunjukkan kemampuan prediksi yang sangat baik pada kelas *dayak* dari sisi precision, namun recall-nya sangat rendah, menandakan model sering memberikan prediksi benar tetapi tidak mampu menangkap seluruh variasi citra kelas tersebut. Arsitektur DenseNet yang menghubungkan setiap layer dengan semua layer sebelumnya sebenarnya sangat efektif dalam pelestarian gradien, tetapi pada dataset kecil, jaringan ini cenderung *overfit* tanpa fine-tuning yang tepat sehingga performanya kurang optimal pada baseline.

Classification Report (EfficientNetB0 - Baseline):

	precision	recall	f1-score	support
balinese	0.7880	0.9355	0.8555	155
batak	0.5000	0.2632	0.3448	19
dayak	0.6667	0.4615	0.5455	13
javanese	0.6596	0.6327	0.6458	49
minangkabau	0.7959	0.6964	0.7429	112
accuracy			0.7615	348
macro avg	0.6820	0.5979	0.6269	348
weighted avg	0.7522	0.7615	0.7502	348

Gambar 3.15 Classification Report Model EfficientNetB0 Baseline

Gambar 3.15 menunjukkan hasil baseline EfficientNetB0 yang memberikan akurasi 76.15% dan macro F1 sebesar 0.6269. Model ini memberikan performa kuat dan stabil, mendekati hasil EfficientNetB3 namun dengan ukuran jaringan yang lebih ringan. EfficientNetB0 mampu mempertahankan precision dan recall tinggi pada kelas mayoritas, dan performanya pada kelas minoritas pun relatif lebih baik dibandingkan model lain seperti DenseNet atau Xception. Hal ini memperlihatkan bahwa EfficientNet tetap efektif bahkan dalam versi yang lebih kecil, menunjukkan efisiensi dan kemampuan generalisasi yang baik tanpa memerlukan arsitektur besar.

```
Classification Report (MobileNetV2 - Baseline):
              precision    recall  f1-score   support

    balinese     0.7753    0.8903    0.8288       155
       batak     0.4444    0.2105    0.2857        19
       dayak     1.0000    0.4615    0.6316        13
    javanese     0.5946    0.4490    0.5116        49
 minangkabau     0.6780    0.7143    0.6957       112

    accuracy                         0.7184       348
   macro avg     0.6985    0.5451    0.5907       348
weighted avg     0.7089    0.7184    0.7043       348
```

Gambar 3.16 Classification Report Model MobileNetV2 Baseline

Gambar 3.16 memperlihatkan hasil baseline MobileNetV2 dengan akurasi sebesar 71.84% dan macro F1 sebesar 0.5907. Sebagai model lightweight yang dirancang untuk perangkat mobile, MobileNetV2 menunjukkan performa yang cukup kompetitif, bahkan mampu mendekati VGG16 dengan ukuran model yang jauh lebih kecil. Performanya bagus pada kelas balinese dan minangkabau, tetapi masih cukup fluktuatif pada kelas minoritas seperti batak dan dayak. Performanya menunjukkan bahwa MobileNetV2 mampu memberikan baseline yang kuat dengan kompleksitas yang rendah, menjadikannya kandidat ideal jika model perlu di-deploy pada lingkungan dengan keterbatasan sumber daya.

Tabel 3.2 Perbandingan tingkat akurasi Baseline Model

No	Model Baseline	Akurasi (%)	Macro F1-score	Keterangan Umum Performa
1	Xception	65	0.33	Performa terendah, gagal mengenali beberapa kelas minoritas
2	DenseNet121	68.39	0.4577	Performa sedang, namun tidak stabil antar kelas
3	VGG16	70.69	0.5945	Stabil dan mampu mengenali hampir semua

No	Model Baseline	Akurasi (%)	Macro F1-score	Keterangan Umum Performa
				kelas
4	MobileNetV2	71.84	0.5907	Efisien dan kompetitif meskipun model ringan
5	EfficientNetB0	76.15	0.6269	Akurasi tinggi dengan arsitektur efisien
6	EfficientNetB3	77	0.66	Performa terbaik dan paling konsisten

Berdasarkan Tabel 3.2, terlihat adanya perbedaan performa yang cukup signifikan antar model baseline yang diuji. Model Xception menunjukkan performa paling rendah dengan akurasi sebesar 65% dan macro F1-score 0.33, yang mengindikasikan ketidakmampuan model dalam mengenali seluruh kelas secara seimbang, khususnya pada kelas minoritas seperti *batak* dan *dayak*. Hal ini menegaskan bahwa penggunaan Xception tanpa fine-tuning belum optimal untuk karakteristik dataset yang digunakan.

Model DenseNet121 mengalami sedikit peningkatan akurasi menjadi 68.39%, namun macro F1-score yang masih relatif rendah (0.4577) menunjukkan bahwa performa model belum stabil antar kelas. Meskipun DenseNet unggul dalam pelestarian gradien, arsitektur ini cenderung kurang optimal pada tahap baseline ketika diterapkan pada dataset berukuran kecil tanpa penyesuaian lanjutan.

Selanjutnya, VGG16 dan MobileNetV2 menunjukkan performa yang lebih stabil dengan akurasi masing-masing 70.69% dan 71.84%. VGG16 unggul dalam konsistensi prediksi berkat arsitekturnya yang sederhana, sementara MobileNetV2 mampu mencapai performa yang hampir setara meskipun memiliki kompleksitas model yang jauh lebih rendah. Hal ini menunjukkan bahwa efisiensi arsitektur tidak selalu

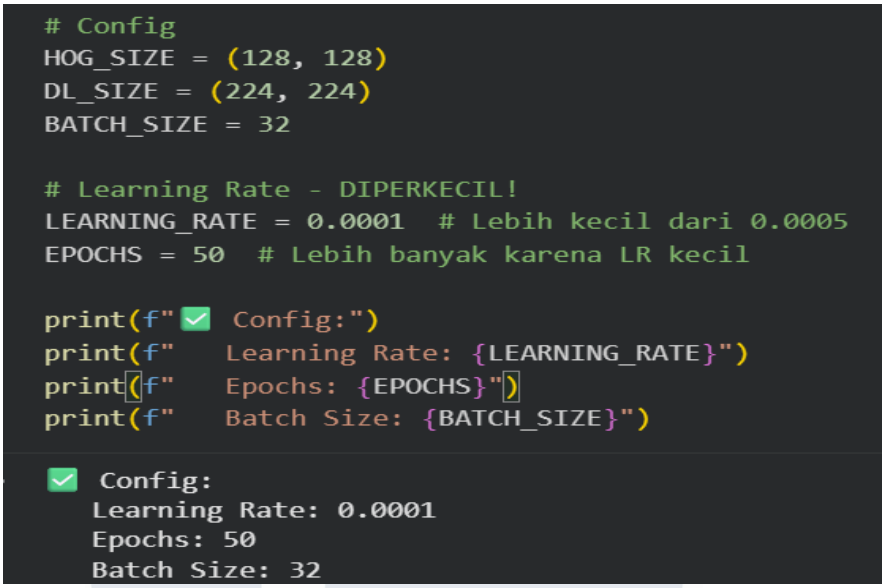
berbanding lurus dengan penurunan performa.

Model EfficientNetB0 dan EfficientNetB3 mencatat performa tertinggi di antara seluruh model baseline. EfficientNetB0 mencapai akurasi 76.15%, sedangkan EfficientNetB3 memperoleh akurasi tertinggi sebesar 77% dengan macro F1-score sekitar 0.66. Keunggulan ini disebabkan oleh pendekatan compound scaling yang memungkinkan model mengekstraksi fitur visual secara lebih efektif dan seimbang, bahkan pada kelas dengan distribusi data yang lebih kecil.

Secara keseluruhan, hasil ini menunjukkan bahwa EfficientNet, khususnya varian B3, merupakan baseline paling kuat dan layak dijadikan fondasi untuk tahap fine-tuning dan pengembangan model lanjutan, sementara model lain berperan sebagai pembanding untuk menilai peningkatan performa yang dicapai.

3.3.1.2 Fine-Tuning dan Penguatan Fitur (HOG = DenseNet121)

Tahap 2 merupakan fase lanjutan setelah pengembangan baseline model, yang berfokus pada peningkatan performa model melalui strategi fine-tuning dan penggabungan fitur menggunakan Histogram of Oriented Gradients (HOG). Berbeda dengan Tahap 1 yang hanya mengevaluasi kemampuan dasar masing-masing arsitektur pretrained, Tahap 2 dirancang untuk mengoptimalkan model terbaik dari baseline dalam hal ini DenseNet121 agar mampu menangkap pola visual yang lebih kaya, khususnya pada kelas-kelas yang sebelumnya memiliki nilai precision dan recall yang rendah. Proses ini mencakup pembukaan sebagian layer pretrained (unfreezing), penurunan learning rate, serta penyatuan deep features dan handcrafted features untuk menghasilkan representasi citra yang lebih komprehensif.



```
# Config
HOG_SIZE = (128, 128)
DL_SIZE = (224, 224)
BATCH_SIZE = 32

# Learning Rate - DIPERKECIL!
LEARNING_RATE = 0.0001  # Lebih kecil dari 0.0005
EPOCHS = 50  # Lebih banyak karena LR kecil

print(f"✅ Config:")
print(f"   Learning Rate: {LEARNING_RATE}")
print(f"   Epochs: {EPOCHS}")
print(f"   Batch Size: {BATCH_SIZE}")
```

```
✅ Config:
   Learning Rate: 0.0001
   Epochs: 50
   Batch Size: 32
```

Gambar 3.17 Konfigurasi Dasar Feature Extraction

Pada Gambar 3.17 ditetapkan konfigurasi dasar untuk pipeline , yaitu ukuran input HOG sebesar 128×128, ukuran input backbone CNN (DenseNet121) sebesar 224×224, batch size 32, serta *learning rate* kecil (0.0001). Penggunaan *learning rate* kecil diperlukan karena model memiliki jumlah fitur yang besar (lebih dari 9.000 fitur gabungan), sehingga pembaruan bobot yang terlalu agresif dapat menyebabkan ketidakstabilan pelatihan. Selain itu, jumlah epoch diperbesar menjadi 50 agar model memiliki waktu pembelajaran yang cukup pada LR kecil tersebut. Bagian ini memastikan seluruh eksperimen berjalan dengan kontrol parameter yang konsisten.

Dataset yang digunakan pada tahap ini telah melalui proses penyeimbangan (balancing) oleh Data Engineer untuk memastikan setiap kelas memiliki jumlah sampel yang proporsional, yaitu 249 citra per kelas. Proses balancing ini penting untuk mengurangi bias model terhadap kelas mayoritas dan memastikan metrik evaluasi seperti Macro F1-score dapat mencerminkan performa model secara adil pada seluruh kelas. Penulis sebagai Data Scientist menerima dataset yang sudah seimbang dan siap

digunakan untuk tahap feature extraction dan pemodelan.

Sebelum melakukan ekstraksi fitur, dilakukan verifikasi visual terhadap sampel dari setiap kelas untuk memastikan kualitas dan keberagaman data. Visualisasi menunjukkan bahwa proses balancing tidak menimbulkan distorsi signifikan dan setiap kelas tetap mempertahankan karakteristik visual yang representatif.

```python
def extract_hog_features(image_path):
    try:
        img = cv2.imread(image_path)
        if img is None:
            return None

        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        resized = cv2.resize(gray, HOG_SIZE)

        features = hog(
            resized,
            orientations=9,
            pixels_per_cell=(8, 8),
            cells_per_block=(2, 2),
            block_norm='L2-Hys',
            visualize=False,
            transform_sqrt=True,
            feature_vector=True
        )
        return features
    except Exception as e:
        return None
```

Gambar 3.18 Ekstrasi HOG

Proses ekstraksi fitur HOG menghasilkan vektor berdimensi 8.100 yang merepresentasikan gradien orientasi pada setiap sel citra. Parameter yang digunakan dalam ekstraksi HOG telah disesuaikan berdasarkan karakteristik visual rumah adat Nusantara. Penggunaan 9 orientasi gradien dipilih karena mampu menangkap variasi arah tepi yang umum ditemukan pada struktur arsitektur tradisional, seperti sudut atap limasan, ukiran vertikal pada tiang, dan pola diagonal pada ornamen. Ukuran sel 8×8 piksel dipilih agar sensitif terhadap detail halus seperti motif ukiran, namun tetap cukup besar untuk menghindari noise dari tekstur material. Normalisasi L2-Hys pada tingkat blok (2×2 sel) memastikan fitur HOG tetap stabil terhadap perubahan pencahayaan yang sering terjadi pada foto rumah adat di berbagai kondisi cuaca dan waktu pengambilan gambar. Fitur HOG ini secara khusus

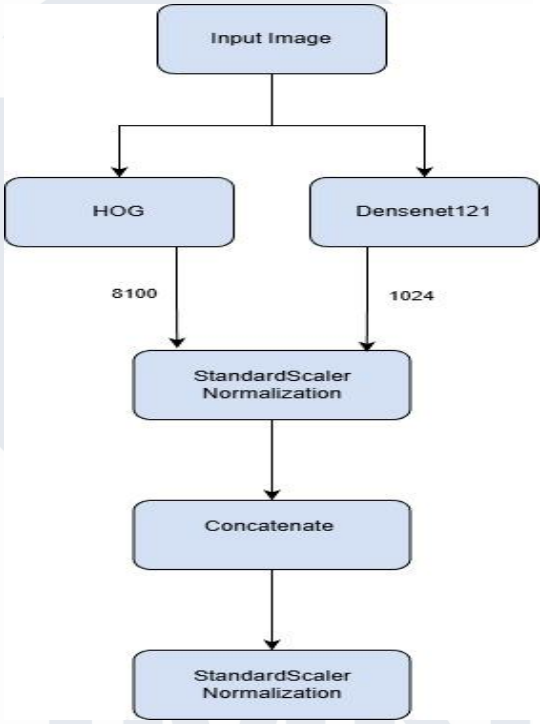
unggul dalam menangkap karakteristik lokal seperti kontur atap, pola ukiran pada dinding, dan detail ornamen yang menjadi ciri khas setiap jenis rumah adat..

```python
base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
feature_extractor = Model(inputs=base_model.input,
                          outputs=GlobalAveragePooling2D()(base_model.output))

print("✅ DenseNet121 feature extractor loaded")

def extract_densenet_features(image_path):
    try:
        img = load_img(image_path, target_size=DL_SIZE)
        img_array = img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)
        features = feature_extractor.predict(img_array, verbose=0)
        return features.flatten()
    except Exception as e:
        return None

print("Extracting DenseNet features...")
densenet_features_list = []

for filepath in tqdm(df_valid['filepath'].values):
    dn_feat = extract_densenet_features(filepath)
    if dn_feat is not None:
        densenet_features_list.append(dn_feat)

densenet_features = np.array(densenet_features_list)
```

Gambar 3.19 Feature Extractor Densenet121

Ekstraksi fitur menggunakan DenseNet121 menghasilkan vektor berdimensi 1.024 yang merepresentasikan abstraksi visual tingkat tinggi. Berbeda dengan HOG yang berfokus pada detail lokal, DenseNet121 menangkap informasi global seperti proporsi bangunan, tata letak ruang, dan hubungan spasial antar elemen arsitektur. Arsitektur DenseNet121 dipilih karena karakteristik dense connectivity-nya yang menghubungkan setiap layer dengan semua layer sebelumnya, sehingga memungkinkan pelestarian fitur dari berbagai tingkat abstraksi. Penggunaan GlobalAveragePooling2D pada output DenseNet121 berfungsi untuk mengkompresi feature maps berdimensi tinggi menjadi vektor fitur yang padat namun tetap mempertahankan informasi penting dari seluruh area citra. Model DenseNet121 yang telah di-pretrain pada dataset ImageNet

membawa pengetahuan tentang struktur visual umum yang kemudian diadaptasi untuk mengenali karakteristik arsitektur rumah adat Nusantara. Fitur yang dihasilkan oleh DenseNet121 secara khusus efektif dalam menangkap bentuk keseluruhan bangunan, komposisi struktural, dan pola visual global yang membedakan satu jenis rumah adat dengan lainnya.



Gambar 3.20 alur feature extraction

Gambar 3.20 menunjukkan alur lengkap proses feature extraction yang diterapkan pada Tahap 2. Setiap citra input terlebih dahulu diproses melalui dua jalur ekstraksi fitur yang independen. Jalur pertama menggunakan HOG untuk menghasilkan fitur berbasis gradien berdimensi 8.100, sementara jalur kedua menggunakan DenseNet121 untuk menghasilkan fitur berbasis deep learning berdimensi 1.024. Kedua vektor fitur kemudian dinormalisasi secara terpisah menggunakan StandardScaler agar memiliki skala yang seragam, kemudian digabungkan menjadi satu vektor fitur berdimensi 9.124. Vektor fitur gabungan ini selanjutnya digunakan sebagai input untuk model classifier berbasis Multilayer Perceptron (MLP) yang terdiri dari empat lapisan dense dengan konfigurasi

512→256→128→5 neuron. Pendekatan ini memastikan bahwa model dapat mempelajari pola visual dari dua perspektif yang berbeda namun saling melengkapi.

```
# Normalize HOG
scaler_hog = StandardScaler()
hog_norm = scaler_hog.fit_transform(hog_features)

# Normalize DenseNet
scaler_densenet = StandardScaler()
densenet_norm = scaler_densenet.fit_transform(densenet_features)

# Combine
X_combined = np.concatenate([hog_norm, densenet_norm], axis=1)
y = df_valid['class'].values

print(f"✅ Features combined")
print(f"   HOG features: {hog_norm.shape[1]}")
print(f"   DenseNet features: {densenet_norm.shape[1]}")
print(f"   Combined features: {X_combined.shape[1]}")
```

```
============================================================
STEP 6: COMBINING FEATURES
============================================================
✅ Features combined
   HOG features: 8100
   DenseNet features: 1024
   Combined features: 9124
```

Gambar 3.21 Combining HOG dan Densenet121

Pada tahap ini, fitur HOG dan DenseNet dinormalisasi secara terpisah menggunakan StandardScaler agar setiap fitur memiliki skala yang seragam. Normalisasi diperlukan karena kedua jenis fitur berasal dari sumber yang berbeda dengan satuan dan rentang nilai yang tidak setara. Setelah dinormalisasi, kedua vektor fitur digabungkan secara horizontal membentuk satu vektor fitur besar per gambar. Kombinasi ini memungkinkan model classifier mendapatkan representasi lengkap yang mencakup baik pola lokal (HOG) maupun pola global (DenseNet). Fusi dua jenis fitur ini merupakan inti dari pendekatan yang digunakan dalam tahap kedua penelitian.

```python
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Stratified split
X_train, X_val, y_train, y_val = train_test_split(
    X_combined,
    y_encoded,
    test_size=0.2,
    stratify=y_encoded,
    random_state=42
)

print(f"✅ Train: {len(X_train)} samples")
print(f"✅ Val: {len(X_val)} samples")
print(f"✅ Classes: {list(label_encoder.classes_)}")

# Verify distribution
print("\n📊 Train class distribution:")
train_dist = np.bincount(y_train)
for i, cls in enumerate(label_encoder.classes_):
    print(f"   {cls:15s}: {train_dist[i]:3d}")

print("\n📊 Val class distribution:")
val_dist = np.bincount(y_val)
for i, cls in enumerate(label_encoder.classes_):
    print(f"   {cls:15s}: {val_dist[i]:3d}")

# Class weights (tetap pakai untuk extra boost)
class_weights = compute_class_weight('balanced', classes=np.unique(y_train), y=y_train)
class_weight_dict = dict(enumerate(class_weights))
```

Gambar 3.22 Encoding Label dan Pemisahan Data

Tahap ini mengonversi label kelas menjadi representasi numerik menggunakan LabelEncoder karena model membutuhkan input berupa angka. Dataset kemudian dibagi secara stratified menjadi data pelatihan dan validasi dengan rasio 80:20 untuk memastikan distribusi kelas tetap merata di kedua subset. Pembagian stratified penting untuk menghindari bias hasil evaluasi. Selain itu, dihitung pula class weight sebagai bentuk kompensasi tambahan selama pelatihan, meskipun dataset sudah diseimbangkan. Penggunaan class weight ini dipertahankan karena tetap dapat membantu model mengatasi variasi kecil dalam distribusi sampel setelah proses sampling.

```
num_classes = len(label_encoder.classes_)

model = Sequential([
    Dense(512, activation='relu', input_shape=(X_combined.shape[1],)),
    BatchNormalization(),
    Dropout(0.5),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.4),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),
    Dense(num_classes, activation='softmax')
])

# Compile dengan LEARNING RATE KECIL
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=LEARNING_RATE),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

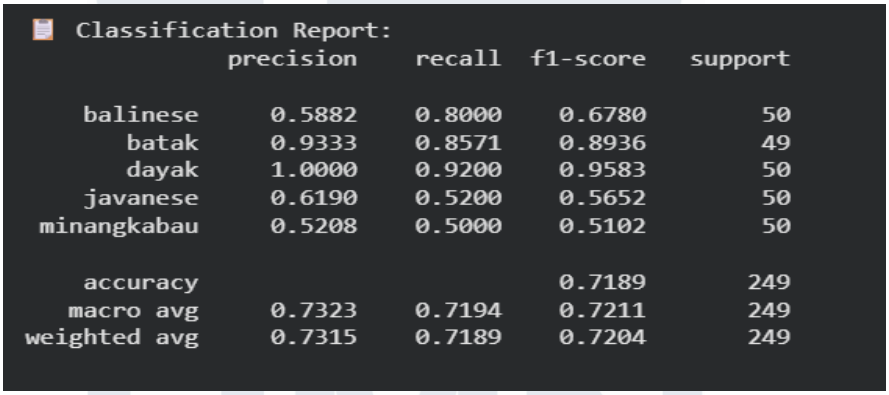
Gambar 3.23 Pembangunan Model Classifier

Model classifier dibangun sebagai jaringan fully-connected (MLP) karena fitur yang digunakan sudah dalam bentuk vektor. Arsitektur terdiri dari tiga lapisan dense berturut-turut sebesar 512, 256, dan 128 neuron, masing-masing disertai BatchNormalization dan Dropout untuk meningkatkan stabilitas dan mencegah overfitting akibat ukuran vektor fitur yang besar. Lapisan akhir menggunakan aktivasi softmax untuk menghasilkan probabilitas per kelas. Optimizer Adam dengan learning rate kecil digunakan untuk memastikan proses pembelajaran terjadi secara stabil dan bertahap. Model ini dirancang untuk mampu mempelajari hubungan antarfitur gabungan yang kompleks.

```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

checkpoint = ModelCheckpoint(
    'best_hybrid_lr_tuned.h5',
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)

# Learning rate reduction - lebih conservative
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,  # Reduce by half
    patience=7,  # Wait longer
    min_lr=1e-8,
    verbose=1
)

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=15,  # More patience karena LR kecil
    restore_best_weights=True,
    verbose=1
)

# Sample weights
sample_weights = np.array([class_weight_dict[label] for label in y_train])
```

Gambar 3.24 Pelatihan Model

Dalam proses pelatihan, digunakan beberapa callback penting seperti ModelCheckpoint, ReduceLROnPlateau, dan EarlyStopping. ModelCheckpoint menyimpan bobot terbaik selama pelatihan berdasarkan nilai akurasi validasi. ReduceLROnPlateau menurunkan learning rate jika model memasuki kondisi stagnan, sehingga membantu pencarian optimum lebih halus. EarlyStopping mencegah pelatihan berlangsung terlalu lama ketika performa tidak lagi meningkat. Sampel pelatihan diberi bobot tambahan melalui sample_weight untuk memperkuat pengaruh tiap kelas. Pelatihan kemudian dilakukan hingga proses early stopping tercapai, sehingga model berhenti pada epoch terbaik dengan performa validasi tertinggi.



Classification Report:

	precision	recall	f1-score	support
balinese	0.5882	0.8000	0.6780	50
batak	0.9333	0.8571	0.8936	49
dayak	1.0000	0.9200	0.9583	50
javanese	0.6190	0.5200	0.5652	50
minangkabau	0.5208	0.5000	0.5102	50
accuracy			0.7189	249
macro avg	0.7323	0.7194	0.7211	249
weighted avg	0.7315	0.7189	0.7204	249

Gambar 3.25 Evaluasi Model HOG dan Densenet

Evaluasi model dilakukan menggunakan data validasi, menghasilkan Macro F1-score sebesar 0.7211, Micro F1-score sebesar 0.7189, dan Weighted F1-score sebesar 0.7204. Ketiga metrik tersebut menunjukkan bahwa model memiliki performa yang konsisten dan mampu menghasilkan prediksi yang relatif seimbang antar kelas. Pencapaian Macro F1 yang cukup tinggi menandakan bahwa pendekatan yang digunakan tidak hanya bekerja baik pada kelas dengan jumlah sampel besar, melainkan juga mempertahankan performa pada kelas dengan distribusi data lebih kecil. Secara umum, nilai-nilai ini mengindikasikan bahwa integrasi antara fitur CNN dan fitur berbasis tekstur serta struktur visual memberikan kontribusi

positif terhadap peningkatan kemampuan model dalam mengenali karakteristik rumah adat yang beragam.

Hasil classification report memperlihatkan bahwa kelas Batak dan Dayak mencapai F1-score yang sangat tinggi, masing-masing sebesar 0.89 dan 0.95. Tingginya performa pada kedua kelas tersebut disebabkan oleh karakter visual rumah adat yang memiliki ciri arsitektural khas dan mudah dibedakan, seperti bentuk atap yang unik, kontur bangunan yang tegas, serta pola struktural yang konsisten. Pendekatan mampu menangkap perbedaan visual semacam ini secara lebih efektif dibandingkan pendekatan CNN murni. Selain itu, model menunjukkan peningkatan stabilitas recall pada kelas yang pada tahap baseline sebelumnya cenderung sulit dikenali, sehingga mengurangi kesalahan pengenalan yang sebelumnya cukup dominan.

Meskipun demikian, performa pada kelas Javanese dan Minangkabau masih berada pada tingkat yang lebih rendah dibanding kelas lainnya. Hal ini berkaitan dengan tingkat variasi arsitektur yang lebih tinggi dalam kedua kelas tersebut, serta adanya kemiripan elemen visual dengan kelas lain, seperti bentuk atap limasan atau kombinasi warna yang tidak terlalu kontras. Variasi semacam ini menyebabkan model kesulitan dalam membentuk representasi fitur yang stabil. Kendati demikian, pendekatan tetap menunjukkan keunggulan karena mampu menangkap pola-pola visual yang bersifat lokal, seperti kontur atap, tekstur material bangunan, dan pola geometrik struktur rumah, sehingga memberikan peningkatan performa keseluruhan yang tidak diperoleh dari model CNN murni.

Secara keseluruhan, hasil evaluasi menunjukkan bahwa penggunaan features memberikan peningkatan performa yang signifikan dibandingkan baseline pada tahap sebelumnya. Pendekatan ini mampu meningkatkan akurasi global dan memperbaiki pemerataan performa antar kelas, menandakan bahwa integrasi antara fitur deep learning dan fitur deskriptif

berbasis tekstur serta bentuk visual merupakan strategi yang lebih efektif untuk tugas klasifikasi rumah adat yang memiliki karakteristik visual kompleks dan beragam.

3.3.1.3 Model Ensemble dan Maksimalisasi Performa

Subbab ini merupakan tahap lanjutan setelah penerapan fine-tuning dan penguatan fitur pada Subbab 3.3.1.2. Berbeda dengan tahap sebelumnya yang hanya menggabungkan fitur HOG dan DenseNet121 dalam satu model tunggal, tahap ini dirancang untuk memaksimalkan performa klasifikasi melalui pendekatan ensemble learning serta fusi fitur tingkat lanjut yang melibatkan tiga sumber fitur, yaitu HOG, DenseNet121, dan EfficientNetB3. Pendekatan ini bertujuan untuk meningkatkan stabilitas prediksi dan pemerataan performa antar kelas rumah adat yang memiliki karakteristik visual beragam.

Strategi ensemble yang diterapkan pada penelitian ini menggunakan metode Bagging (Bootstrap Aggregating) dengan variasi hyperparameter. Bagging adalah teknik ensemble learning yang melatih multiple models secara parallel dan independen, kemudian menggabungkan prediksi menggunakan averaging untuk mengurangi variance dan meningkatkan stabilitas prediksi. Penelitian ini mengadopsi prinsip fundamental bagging yaitu parallel training dan simple averaging, namun dengan modifikasi pada sumber diversity.

```
# Config
HOG_SIZE = (128, 128)
DL_SIZE = (224, 224)
BATCH_SIZE = 32

# MULTIPLE LEARNING RATES untuk ensemble
LEARNING_RATES = [0.0001, 0.00005]  # Train 2 model dengan LR berbeda
EPOCHS = 60  # Lebih banyak
```

Gambar 3.26 Konfigurasi Ensemble

Gambar 3.26 menunjukkan konfigurasi parameter utama yang digunakan pada tahap ensemble. Pada Subbab 3.3.1.2, proses pelatihan hanya menggunakan satu nilai learning rate sebesar 0.0001. Namun, pada Subbab 3.3.1.3 nilai learning rate diperluas menjadi dua variasi, yaitu 0.0001 dan 0.00005. Perubahan ini dilakukan dengan tujuan menghasilkan beberapa model dengan karakteristik konvergensi yang berbeda. Learning rate yang lebih besar memungkinkan model beradaptasi lebih cepat pada awal pelatihan, sementara learning rate yang lebih kecil mendorong proses pembelajaran yang lebih halus dan stabil pada fase lanjutan.

Konfigurasi ensemble pada Gambar 3.29 menunjukkan penggunaan dua variasi learning rate (0.0001 dan 0.00005) sebagai sumber diversity. Dua model Multilayer Perceptron (MLP) dengan arsitektur identik dilatih secara parallel pada dataset yang sama, namun dengan learning rate berbeda. Model pertama dengan learning rate 0.0001 melakukan eksplorasi parameter space lebih luas dan mencapai konvergensi lebih cepat (sekitar 35-40 epoch), cenderung lebih generalis dalam menangkap pola umum. Model kedua dengan learning rate 0.00005 melakukan pembelajaran lebih teliti dan mencapai konvergensi lebih lambat (sekitar 45-50 epoch), cenderung lebih spesialis dalam menangkap detail dan nuansa. Perbedaan karakteristik pembelajaran ini menciptakan diversity yang diperlukan untuk efektivitas ensemble.

Perbedaan dengan bagging tradisional terletak pada sumber diversity. Bagging tradisional (seperti Random Forest) menciptakan diversity melalui bootstrap sampling, di mana setiap model dilatih pada subset data berbeda yang diambil secara acak dengan penggantian (sampling with replacement), sehingga setiap model hanya melihat sekitar 63% data unik dan sisanya adalah duplikasi atau tidak terlihat sama sekali. Pendekatan dalam penelitian ini menciptakan diversity melalui variasi learning rate, sehingga setiap model dilatih pada seluruh data (100%) tanpa ada data yang terbuang. Pendekatan ini dipilih karena dataset yang sudah

di-balancing memiliki jumlah sampel terbatas (249 per kelas), sehingga memanfaatkan seluruh data untuk setiap model lebih optimal dibandingkan membagi data melalui bootstrap sampling.

Jumlah epoch ditingkatkan menjadi 60 untuk memberikan ruang pembelajaran yang cukup bagi model dengan learning rate kecil agar dapat mencapai titik konvergensi optimal. Batch size dipertahankan sebesar 32 untuk menjaga keseimbangan antara stabilitas gradien dan efisiensi komputasi. Output dari tahap ini berupa konfigurasi eksperimen yang menjadi dasar seluruh proses pelatihan ensemble.

```
        features = hog(
            resized,
            orientations=9,
            pixels_per_cell=(8, 8),
            cells_per_block=(2, 2),
            block_norm='L2-Hys',
            visualize=False,
            transform_sqrt=True,
            feature_vector=True
        )
        return features
    except:
        return None
```

Gambar 3.27 Ekstraksi Fitur Histogram of Oriented Gradients (HOG)

Gambar 3.27 menunjukkan tahap ekstraksi fitur HOG yang bertujuan menangkap karakteristik tekstur dan struktur lokal citra rumah adat. Setiap citra diubah menjadi grayscale dan diskalakan ke ukuran yang telah ditentukan, kemudian dihitung gradien orientasinya. Parameter orientasi sebanyak sembilan arah dipilih untuk merepresentasikan variasi arah tepi yang umum ditemukan pada pola arsitektur rumah adat.

Penggunaan HOG pada tahap ini konsisten dengan Subbab 3.3.1.2, namun perannya menjadi lebih krusial karena HOG berfungsi sebagai pelengkap bagi dua backbone CNN yang digunakan. Output dari tahap ini berupa vektor fitur berdimensi tinggi yang merepresentasikan pola tepi,

kontur atap, ukiran kayu, dan struktur geometrik rumah adat, yang sering kali tidak tertangkap secara optimal oleh CNN murni.

```
base_densenet = DenseNet121(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
densenet_extractor = Model(inputs=base_densenet.input,
                           outputs=GlobalAveragePooling2D()(base_densenet.output))
```

Gambar 3.28 Ekstraksi Fitur Global
Menggunakan DenseNet121

Gambar 3.28 menampilkan proses ekstraksi fitur menggunakan DenseNet121 sebagai backbone CNN utama. DenseNet121 dipilih karena arsitekturnya yang menggunakan dense connectivity, sehingga memungkinkan fitur dari layer awal dan akhir saling terhubung dan memperkaya representasi fitur. Pada tahap ini, DenseNet121 digunakan sebagai feature extractor tanpa lapisan klasifikasi akhir.

Berbeda dengan Subbab 3.3.1.2 yang hanya mengandalkan DenseNet121 sebagai satu-satunya CNN, pada Subbab 3.3.1.3 DenseNet121 berperan sebagai salah satu sumber fitur global. Output yang dihasilkan berupa vektor fitur berdimensi 1024 yang menggambarkan struktur bangunan, komposisi visual, dan pola global rumah adat.

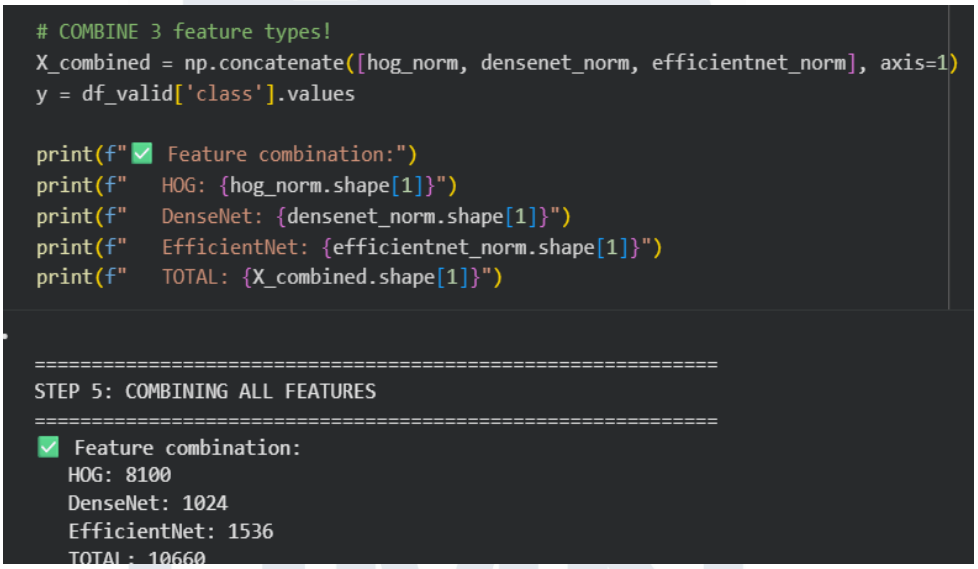
```
base_efficientnet = EfficientNetB3(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
efficientnet_extractor = Model(inputs=base_efficientnet.input,
                               outputs=GlobalAveragePooling2D()(base_efficientnet.output))
```

Gambar 3.29 Ekstraksi Fitur Multi-Skala
Menggunakan EfficientNetB3

Gambar 3.29 menunjukkan proses ekstraksi fitur menggunakan EfficientNetB3. Penambahan EfficientNetB3 merupakan perbedaan utama dibandingkan Subbab 3.3.1.2. EfficientNetB3 dipilih karena kemampuannya menangkap pola visual multi-skala melalui mekanisme

compound scaling, sehingga efektif dalam mengenali detail halus dan variasi ukuran objek.

Output dari tahap ini berupa vektor fitur berdimensi 1536 yang mengandung informasi visual tingkat lanjut, termasuk variasi warna lokal, detail ornamen, dan pola arsitektur yang lebih kompleks. Kehadiran EfficientNetB3 melengkapi DenseNet121 yang lebih fokus pada struktur global, sehingga memperkaya representasi fitur secara keseluruhan.



Gambar 3.30 Normalisasi dan Fusi Tiga Jenis Fitur

Gambar 3.30 menggambarkan tahap normalisasi dan penggabungan fitur HOG, DenseNet121, dan EfficientNetB3. Sebelum digabungkan, seluruh fitur dinormalisasi untuk menyamakan rentang nilai, mengingat perbedaan karakteristik dan skala antar fitur. Tahap ini merupakan inti dari pendekatan -ensemble yang digunakan.

Output tahap ini berupa vektor fitur berdimensi sangat tinggi yang menggabungkan informasi tekstur lokal, struktur global, dan pola multi-skala. Representasi ini memungkinkan model klasifikasi untuk mempelajari hubungan kompleks antar fitur yang sebelumnya tidak tersedia pada pendekatan tunggal.

```
X_train, X_val, y_train, y_val = train_test_split(
    X_combined, y_encoded,
    test_size=0.2,
    stratify=y_encoded,
    random_state=42
)
```

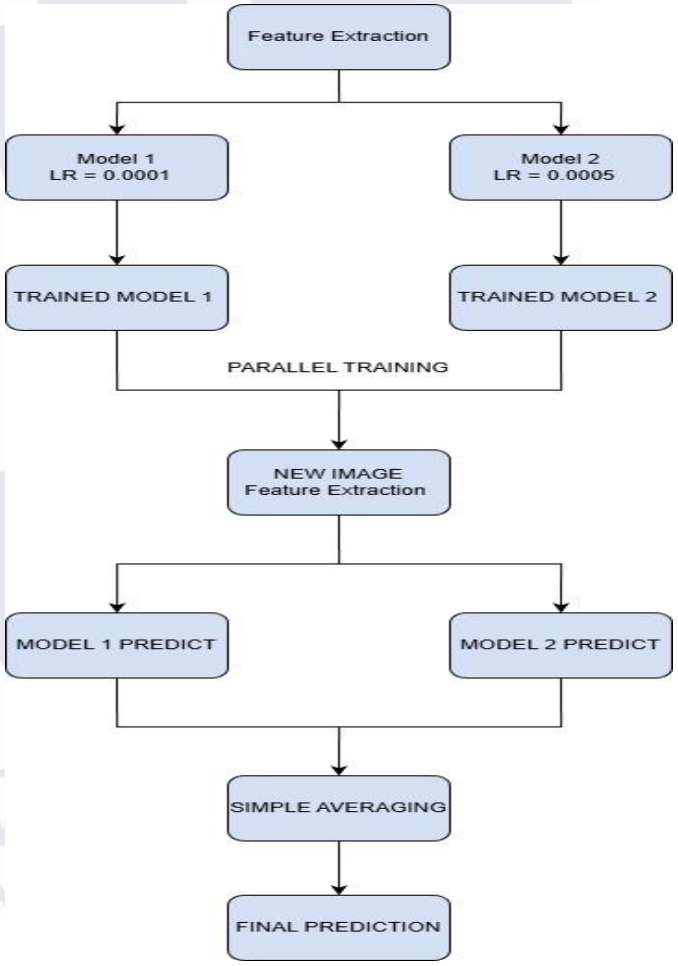
Gambar 3.31 Pembagian Data dan Penentuan Bobot Kelas

Proses pembagian data pada Gambar 3.31 sama dengan Subbab 3.3.1.2, yaitu menggunakan pembagian stratified dengan rasio 80:20. Strategi ini dipertahankan karena terbukti mampu menjaga distribusi kelas secara konsisten antara data latih dan validasi. Bobot kelas yang dihasilkan relatif mendekati satu, menunjukkan bahwa proses balancing sebelumnya telah berjalan dengan baik dan tidak menimbulkan dominasi kelas tertentu.

```
def build_model(learning_rate):
    model = Sequential([
        Dense(1024, activation='relu', input_shape=(X_combined.shape[1],)),
        BatchNormalization(),
        Dropout(0.5),

        Dense(512, activation='relu'),
        BatchNormalization(),
        Dropout(0.5),

        Dense(256, activation='relu'),
        BatchNormalization(),
        Dropout(0.4),

        Dense(128, activation='relu'),
        BatchNormalization(),
        Dropout(0.3),

        Dense(num_classes, activation='softmax')
    ])

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )

    return model
```

Gambar 3.32 Perancangan Arsitektur Model Klasifikasi

Gambar 3.32 memperlihatkan arsitektur model klasifikasi berbasis multilayer perceptron. Jumlah neuron pada lapisan awal diperbesar dibandingkan Subbab 3.3.1.2 untuk mengakomodasi dimensi fitur gabungan yang lebih besar. Penggunaan beberapa lapisan dense bertujuan melakukan proses reduksi dimensi secara bertahap sekaligus mempertahankan informasi penting dari fitur gabungan. Output tahap ini berupa model klasifikasi yang siap untuk dilatih menggunakan fitur - ensemble.



Gambar 3.33 Ensemble Flowchart

Gambar 3.33 mengilustrasikan mekanisme bagging dengan variasi learning rate yang diterapkan dalam penelitian ini. Bagging (Bootstrap Aggregating) adalah teknik ensemble learning yang melatih beberapa model

secara parallel dan independen, kemudian menggabungkan prediksinya menggunakan averaging untuk mengurangi variasi dan meningkatkan stabilitas. Penelitian ini mengadopsi prinsip fundamental bagging yaitu parallel training dan simple averaging, namun dengan modifikasi pada sumber keberagaman: alih-alih menggunakan bootstrap sampling seperti bagging tradisional (yang membuat setiap model dilatih pada sekitar 63% data berbeda), penelitian ini menggunakan variasi learning rate sehingga kedua model dilatih pada seluruh data (100%) sambil tetap menciptakan keberagaman melalui karakteristik pembelajaran yang berbeda.

Pada Training Phase, dua model Multilayer Perceptron dengan arsitektur identik dilatih secara parallel pada dataset yang sama (996 samples) dengan learning rate berbeda. Model pertama (LR=0.0001) melakukan eksplorasi luas dan konvergensi cepat sekitar 35 epoch, cenderung generalis. Model kedua (LR=0.00005) melakukan pembelajaran teliti dan konvergensi lambat sekitar 45 epoch, cenderung spesialis. Kombinasi kedua karakteristik ini menciptakan saling melengkapi yang efektif: model generalis memberikan kepercayaan diri melalui eksplorasi luas, sementara model spesialis memberikan presisi melalui pembelajaran teliti.

Pada Prediction Phase, citra baru diekstraksi fiturnya menjadi vektor Fitur 10.660 dimensi (HOG + DenseNet121 + EfficientNetB3), kemudian diinputkan ke kedua model menghasilkan dua vektor probabilitas. Probabilitas dari kedua model dirata-ratakan menggunakan simple averaging sesuai prinsip bagging:

$$\boldsymbol{P\_ensemble}(\boldsymbol{c}_{\mathrm{i}}) \;=\; [\boldsymbol{P}_1(\boldsymbol{c}_{\mathrm{i}}) \;+\; \boldsymbol{P}_2(\boldsymbol{c}_{\mathrm{i}})] \,/\, \mathbf{2}$$

Kelas dengan probabilitas ensemble tertinggi dipilih sebagai prediksi akhir menggunakan operasi argmax. Simple averaging mengurangi ketidakstabilan prediksi dan menghasilkan hasil yang lebih konsisten dibandingkan model tunggal.

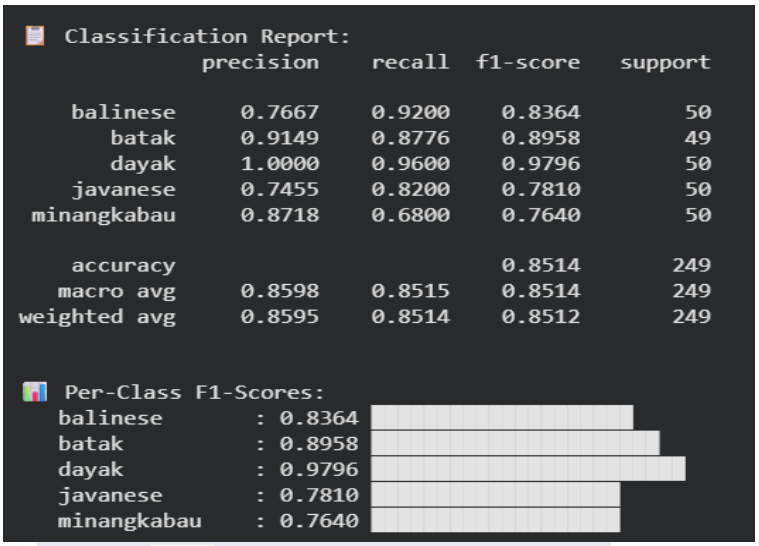
```
checkpoint = ModelCheckpoint(
    f'best_model_lr{lr}.h5',
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)

reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=8,
    min_lr=1e-8,
    verbose=1
)

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=15,
    restore_best_weights=True,
    verbose=1
)
```

Gambar 3.34 Pelatihan Model Ensemble

Gambar 3.38 menunjukkan proses pelatihan beberapa model klasifikasi dalam skema ensemble dengan variasi learning rate. Setiap model dilatih menggunakan arsitektur yang sama dan bobot sampel untuk mengatasi ketidakseimbangan kelas, namun dengan nilai learning rate berbeda guna menghasilkan pola pembelajaran yang beragam. Strategi ini bertujuan meningkatkan kemampuan generalisasi dan mengurangi ketergantungan pada satu model tunggal.

Selama pelatihan, digunakan mekanisme pengendalian pembelajaran berupa *ModelCheckpoint*, *ReduceLROnPlateau*, dan *EarlyStopping* untuk memastikan proses pelatihan berlangsung stabil serta terhindar dari overfitting. Hasil dari tahap ini adalah sekumpulan model terlatih yang siap digabungkan pada tahap ensemble prediction, yang terbukti memberikan peningkatan performa evaluasi, khususnya pada nilai

Macro F1-score.



Gambar 3.35 Hasil Evaluasi Model Ensemble

Hasil evaluasi menunjukkan bahwa pendekatan ensemble menghasilkan performa klasifikasi yang tinggi dan stabil pada seluruh kelas rumah adat. Nilai Macro F1-score sebesar 0.8514 menandakan bahwa model mampu mempertahankan kinerja yang relatif seimbang di seluruh kelas tanpa bias terhadap kelas tertentu. Kesetaraan nilai Micro F1-score (0.8514) dan Weighted F1-score (0.8512) mengindikasikan bahwa distribusi prediksi model konsisten dengan distribusi data validasi, sehingga performa global model tidak didominasi oleh kelas mayoritas.

Berdasarkan *classification report*, kelas Dayak mencapai performa tertinggi dengan F1-score sebesar 0.9796, yang menunjukkan bahwa karakteristik visual rumah adat Dayak memiliki pola yang sangat khas dan mudah dibedakan oleh model ensemble. Kelas Batak juga menunjukkan performa tinggi dengan F1-score 0.8958, mencerminkan keberhasilan model dalam menangkap struktur arsitektural yang kuat dan konsisten. Kelas Balinese memperlihatkan nilai recall yang tinggi (0.9200), menandakan kemampuan model dalam mengenali sebagian besar sampel kelas tersebut, meskipun precision masih relatif lebih rendah akibat

kemiripan visual dengan kelas lain.

Sementara itu, kelas Javanese dan Minangkabau menunjukkan nilai F1-score yang lebih rendah dibandingkan kelas lainnya, masing-masing sebesar 0.7810 dan 0.7640. Hal ini disebabkan oleh tingginya variasi arsitektur serta kemiripan elemen visual antarkelas, seperti bentuk atap dan komposisi warna, yang meningkatkan potensi kesalahan klasifikasi. Meskipun demikian, performa kedua kelas tersebut tetap mengalami peningkatan dibandingkan model baseline, menunjukkan bahwa pendekatan ensemble berbasis fusi fitur HOG, DenseNet121, dan EfficientNetB3 mampu menangkap pola visual yang lebih kaya dan kompleks.

Jika dibandingkan dengan hasil pada Tahap 3.3.1.2 yang menggunakan pendekatan HOG dan DenseNet121, penerapan strategi ensemble pada Tahap 3.3.1.3 menunjukkan peningkatan performa yang signifikan dan lebih merata. Pada Tahap 3.3.1.2, model memperoleh Macro F1-score sebesar 0.7211, yang mengindikasikan bahwa meskipun terjadi peningkatan dibandingkan baseline CNN, performa antar kelas masih belum sepenuhnya seimbang. Kelas Javanese dan Minangkabau pada tahap ini masih menunjukkan nilai F1-score yang relatif rendah, masing-masing sebesar 0.5652 dan 0.5102, sehingga kontribusi kelas minoritas terhadap performa keseluruhan model masih terbatas.

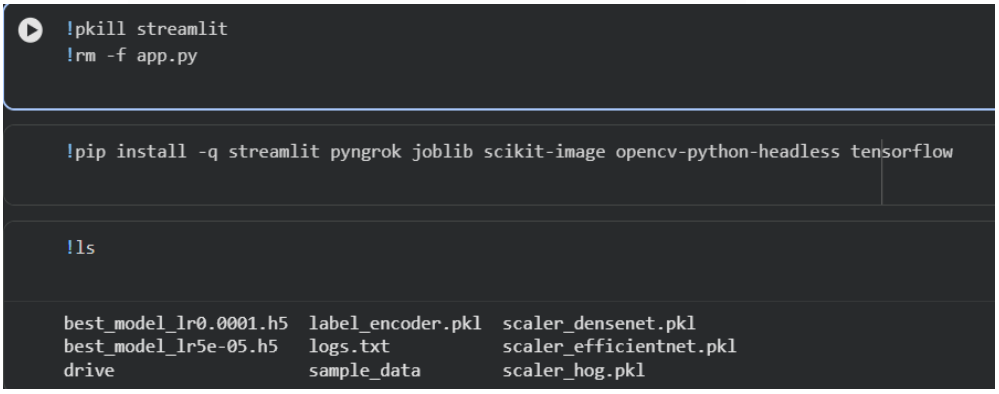
Sebaliknya, hasil evaluasi pada Tahap 3.3.1.3 menunjukkan peningkatan Macro F1-score menjadi 0.8514, yang menandakan perbaikan signifikan dalam pemerataan performa lintas kelas. Peningkatan paling menonjol terjadi pada kelas Javanese dan Minangkabau, yang F1-score-nya meningkat menjadi 0.7810 dan 0.7640. Hal ini menunjukkan bahwa pendekatan ensemble mampu mengurangi kesalahan klasifikasi pada kelas dengan variasi arsitektur tinggi dan kemiripan visual antarkelas. Selain itu, kelas Dayak dan Batak tetap mempertahankan performa sangat tinggi,

masing-masing dengan F1-score 0.9796 dan 0.8958, yang menunjukkan bahwa peningkatan performa tidak mengorbankan kelas yang telah memiliki kinerja baik.

Secara keseluruhan, perbandingan ini menegaskan bahwa strategi ensemble memberikan keuntungan yang jelas dibandingkan pendekatan tunggal pada Tahap 3.3.1.2. Dengan menggabungkan prediksi dari beberapa model yang memiliki karakteristik pembelajaran berbeda, sistem klasifikasi menjadi lebih robust dan stabil, serta mampu melakukan generalisasi yang lebih baik terhadap keragaman visual rumah adat. Temuan ini memperkuat kesimpulan bahwa ensemble merupakan pendekatan yang efektif untuk meningkatkan performa klasifikasi pada domain citra dengan kompleksitas visual tinggi.

3.3.1.4 Tahap Deployment

Tahap deployment merupakan tahap akhir dalam rangkaian pengembangan sistem klasifikasi citra budaya nusantarasetelah diperoleh model ensemble dengan performa terbaik pada Subbab 3.3.1.3. Tahap ini bertujuan untuk mengimplementasikan model klasifikasi ke dalam sebuah lingkungan aplikasi sehingga dapat digunakan secara langsung untuk melakukan prediksi terhadap citra baru di luar data pelatihan dan validasi. Dengan demikian, evaluasi tidak hanya dilakukan secara numerik, tetapi juga melalui pengujian fungsional sistem dalam skenario penggunaan nyata.



Gambar 3.36 Intstalasi dan verifikasi model

Bagian kode pada gambar 3.40 ini digunakan untuk menyiapkan lingkungan eksekusi sebelum aplikasi dijalankan. Perintah penghentian proses Streamlit (pkill streamlit) dan penghapusan berkas aplikasi sebelumnya (rm app.py) bertujuan memastikan tidak terdapat konflik proses maupun file lama yang dapat memengaruhi jalannya aplikasi. Selanjutnya dilakukan instalasi pustaka utama seperti *Streamlit* untuk antarmuka web, *TensorFlow* untuk pemuatan model deep learning, *scikit-image* dan *OpenCV* untuk ekstraksi fitur citra, serta *joblib* untuk memuat objek hasil pelatihan seperti scaler dan label encoder. Output dari tahap ini berupa lingkungan runtime yang bersih dan telah memiliki seluruh dependensi yang diperlukan untuk proses deployment.

Bagian kode yang menampilkan daftar file (ls) berfungsi untuk memastikan bahwa seluruh artefak hasil pelatihan tersedia sebelum aplikasi dijalankan. File yang diverifikasi meliputi dua model ensemble terbaik (best_model_lr0.0001.h5 dan best_model_lr5e-05.h5), scaler untuk masing-masing jenis fitur (HOG, DenseNet121, dan EfficientNetB3), serta label encoder. Keberadaan file-file ini menjadi prasyarat utama agar proses inferensi pada tahap deployment dapat berjalan konsisten dengan proses pelatihan. Output tahap ini berupa konfirmasi bahwa seluruh komponen sistem inferensi telah tersedia secara lengkap.

```
# ================ FEATURE EXTRACTOR ================
base_dn = DenseNet121(weights="imagenet", include_top=False, input_shape=(224,224,3))
dn_extractor = Model(base_dn.input, GlobalAveragePooling2D()(base_dn.output))

base_eff = EfficientNetB3(weights="imagenet", include_top=False, input_shape=(224,224,3))
eff_extractor = Model(base_eff.input, GlobalAveragePooling2D()(base_eff.output))

# ================ FEATURE FUNCTIONS ================
def hog_feature(img):
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    gray = cv2.resize(gray, HOG_SIZE)
    return hog(gray, orientations=9, pixels_per_cell=(8,8),
               cells_per_block=(2,2), block_norm="L2-Hys")

def dn_feature(img):
    img = cv2.resize(img, DL_SIZE) / 255.0
    img = np.expand_dims(img, axis=0)
    return dn_extractor.predict(img, verbose=0).flatten()

def eff_feature(img):
    img = cv2.resize(img, DL_SIZE) / 255.0
    img = np.expand_dims(img, axis=0)
    return eff_extractor.predict(img, verbose=0).flatten()
```

Gambar 3.37 Implementasi Pipeline Ekstraksi Fitur

Bagian kode ini menjelaskan proses ekstraksi fitur dari citra input. Fitur HOG digunakan untuk menangkap tekstur dan struktur lokal seperti tepi, kontur, dan pola geometrik. DenseNet121 berperan mengekstraksi fitur global yang merepresentasikan struktur dan komposisi visual bangunan, sedangkan EfficientNetB3 mengekstraksi fitur multi-skala yang sensitif terhadap detail halus dan variasi ukuran objek. Seluruh fitur yang dihasilkan kemudian dinormalisasi menggunakan scaler yang sama dengan tahap pelatihan, lalu digabungkan menjadi satu vektor fitur. Output tahap ini berupa representasi fitur citra yang siap diproses oleh model ensemble.

Pada tahap ini, vektor fitur hasil fusi dimasukkan ke dalam dua model ensemble. Masing-masing model menghasilkan probabilitas prediksi kelas, yang kemudian digabungkan menggunakan metode rata-rata probabilitas. Pendekatan ini bertujuan meningkatkan stabilitas prediksi dan mengurangi kesalahan individual model. Hasil prediksi akhir ditentukan berdasarkan probabilitas tertinggi, kemudian dikonversi kembali ke label kelas rumah adat menggunakan label encoder. Selain label prediksi, sistem juga menampilkan nilai confidence sebagai ukuran tingkat keyakinan model. Output dari tahap ini berupa hasil klasifikasi rumah adat yang dapat dipahami langsung oleh pengguna.

```
st.set_page_config("Klasifikasi Rumah Adat", layout="centered")
st.title("🏠 Klasifikasi Rumah Adat Nusantara")
st.caption("HOG + DenseNet121 + EfficientNetB3 | Ensemble Learning")

file = st.file_uploader("Upload gambar rumah adat", type=["jpg","png","jpeg"])

if file:
    image = Image.open(file).convert("RGB")
    st.image(image, use_column_width=True)
    img = np.array(image)

    hog_f = scaler_hog.transform([hog_feature(img)])
    dn_f = scaler_dn.transform([dn_feature(img)])
    eff_f = scaler_eff.transform([eff_feature(img)])

    X = np.concatenate([hog_f, dn_f, eff_f], axis=1)

    p1 = model1.predict(X)
    p2 = model2.predict(X)
    pred = (p1 + p2) / 2

    label = label_encoder.inverse_transform([np.argmax(pred)])[0]
    conf = np.max(pred) * 100

    st.success(f"Prediksi: {label}")
    st.info(f"Confidence: {conf:.2f}%")
```

Gambar 3.38 Pembuatan Aplikasi Streamlit dan Pemuatan Model

Gambar 3.38 menunjukkan tampilan antarmuka aplikasi klasifikasi rumah adat yang dikembangkan. Antarmuka ini menyediakan fitur unggah citra, di mana pengguna dapat memasukkan citra rumah adat dalam format umum seperti JPG atau PNG. Setelah citra berhasil diunggah, sistem akan menampilkan pratinjau citra sebagai bentuk umpan balik visual kepada pengguna sebelum proses klasifikasi dilakukan.

Antarmuka dirancang dengan pendekatan minimalis agar pengguna dapat berfokus pada fungsi utama aplikasi, yaitu pengenalan jenis rumah adat berdasarkan citra yang diunggah.

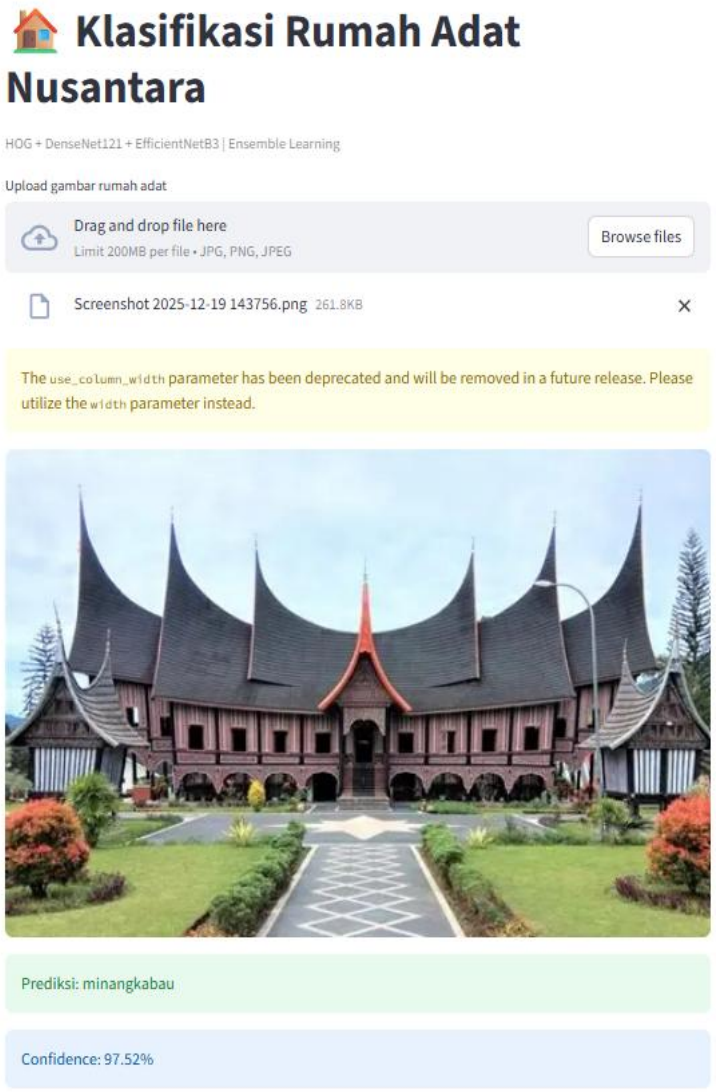
Gambar 3.39 Publikasi Aplikasi Menggunakan Ngrok

Bagian kode pada gambar 3.39 terakhir digunakan untuk mempublikasikan aplikasi Streamlit agar dapat diakses secara daring. Streamlit dijalankan pada port lokal, kemudian Ngrok digunakan untuk membuat *secure tunnel* yang menghasilkan URL publik. Tujuan penggunaan Ngrok adalah memungkinkan demonstrasi dan pengujian sistem secara real-time tanpa memerlukan server produksi. Output dari tahap ini berupa tautan publik yang dapat diakses melalui browser, menandakan bahwa sistem klasifikasi rumah adat telah berhasil dideploy dan siap digunakan oleh pengguna akhir.

Gambar 3. 40 Tampilan Halaman

Gambar 3.40 menampilkan halaman utama aplikasi klasifikasi rumah adat Nusantara. Pada bagian atas halaman ditampilkan judul aplikasi yang berfungsi sebagai identitas sistem, diikuti dengan deskripsi singkat yang menjelaskan metode utama yang digunakan, yaitu pendekatan feature extraction (HOG, DenseNet121, dan EfficientNetB3) serta ensemble learning. Informasi ini bertujuan memberikan konteks awal kepada pengguna mengenai kemampuan dan cakupan sistem sebelum melakukan interaksi lebih lanjut.

komponen unggah citra yang menjadi elemen utama pada antarmuka aplikasi. Komponen ini memungkinkan pengguna untuk memasukkan citra rumah adat dari perangkat lokal dalam format umum seperti JPG, JPEG, atau PNG. Pemilihan mekanisme unggah citra dilakukan untuk memastikan fleksibilitas penggunaan serta kompatibilitas dengan berbagai sumber citra.



Gambar 3.41 Hasil Prediksi

Gambar 3.41 menggambarkan tampilan status sistem selama proses klasifikasi berlangsung. Setelah citra diunggah, sistem secara otomatis menjalankan seluruh tahapan inferensi, mulai dari ekstraksi fitur hingga prediksi kelas rumah adat. Selama proses ini, antarmuka memberikan indikasi bahwa sistem sedang memproses data, sehingga pengguna memperoleh kejelasan bahwa aplikasi berfungsi secara aktif.

Pada gambar 3.45 dapat dilihat setelah melakukan pengunggahan gambar, maka sistem akan secara otomatim menampilkan gambar, hasil prediksi, dan tingkat confidence score dari gambar.

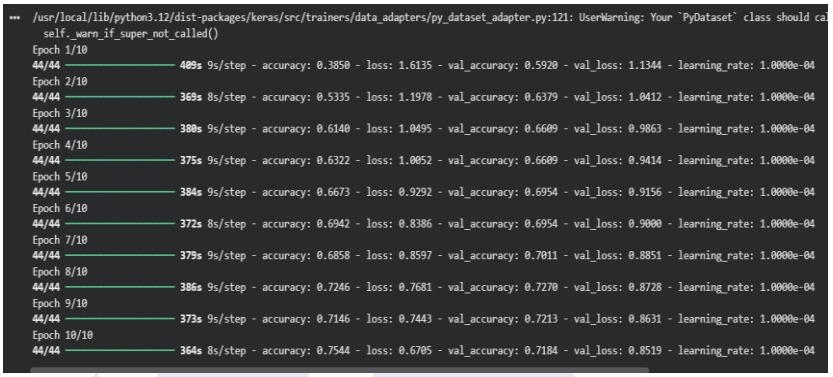
3.1.1 Kendala yang Ditemukan

Selama proses pelaksanaan pengembangan model dalam program PRO-STEP: Road to Champion, penulis menghadapi sejumlah kendala teknis yang signifikan pada tahap pemodelan dan pelatihan model deep learning untuk klasifikasi citra rumah adat Nusantara. Kendala-kendala ini tidak hanya berdampak pada efisiensi waktu pengembangan, tetapi juga mempengaruhi kualitas dan stabilitas performa model yang dihasilkan. Setiap kendala yang dihadapi memberikan pembelajaran penting mengenai kompleksitas pengembangan sistem machine learning dalam konteks data dengan karakteristik visual yang beragam dan dengan keterbatasan sumber daya.

1) **Keterbatasan sumber daya komputasi.**

Keterbatasan sumber daya komputasi menjadi salah satu kendala utama dalam pelaksanaan penelitian ini, terutama pada tahap pelatihan model deep learning. Seluruh proses training dilakukan menggunakan Google Colab dengan alokasi GPU terbatas, sehingga waktu eksekusi pelatihan tetap menjadi faktor pembatas, meskipun digunakan arsitektur yang relatif ringan conothnya seperti MobileNetV2.

Berdasarkan hasil eksperimen, pelatihan MobileNetV2 selama 10 epoch membutuhkan waktu sekitar 360–410 detik per epoch, dengan rata-rata 8–9 detik per step untuk 44 step pada setiap epoch. Dengan demikian, total waktu pelatihan satu model MobileNetV2 mencapai sekitar 60–70 menit. Dibandingkan dengan arsitektur yang lebih kompleks, MobileNetV2 menunjukkan efisiensi komputasi yang lebih baik, namun tetap memerlukan waktu yang signifikan dalam konteks keterbatasan resource.

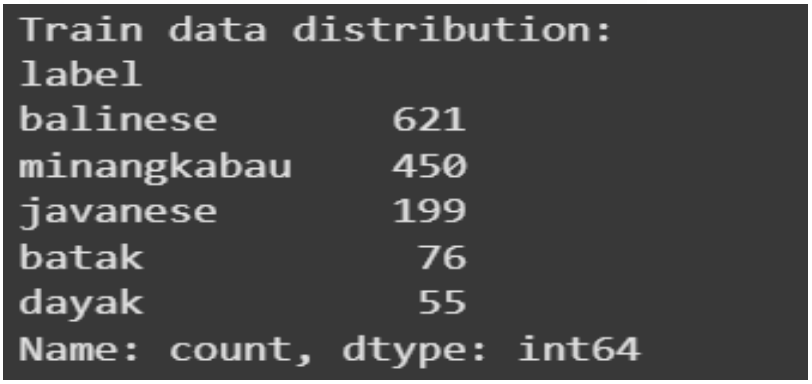


Gambar 3.42 epoch pelatihan MobileNetV2

Proses pelatihan MobileNetV2 pada Google Colab menunjukkan waktu eksekusi sekitar 360–410 detik per epoch dengan 44 step. Meskipun MobileNetV2 dirancang sebagai arsitektur lightweight, keterbatasan hardware tetap memberikan dampak teknis yang nyata.

2) **Dataset besar dan tidak seimbang (imbalanced data).**

Salah satu kendala utama yang penulis temui selama proses pengerjaan proyek ini adalah ketidakseimbangan jumlah data (*data imbalanced*) antar kelas pada dataset citra rumah adat Indonesia. Berdasarkan hasil eksplorasi awal, ditemukan bahwa jumlah citra pada setiap kelas tidak merata, di mana kelas *Balinese* dan *Minangkabau* memiliki jumlah data yang jauh lebih banyak dibandingkan dengan kelas *Batak* dan *Dayak*.



Gambar 3.43 Imbalanced Data

Distribusi data pelatihan menunjukkan perbedaan

yang cukup signifikan, yaitu *Balinese* sebanyak 621 gambar, *Minangkabau* 450 gambar, *Javanese* 199 gambar, *Batak* 76 gambar, dan *Dayak* hanya 55 gambar. Ketidakseimbangan ini juga terlihat pada data validasi dengan proporsi yang serupa. Kondisi tersebut menjadi kendala karena model cenderung belajar lebih dominan dari kelas dengan data terbanyak, sehingga menghasilkan prediksi yang bias terhadap kelas mayoritas dan akurasi rendah pada kelas minoritas.

Kendala ini berdampak pada proses pelatihan yang memerlukan waktu lebih lama untuk mencapai *convergence*, serta hasil evaluasi model yang menunjukkan ketidakseimbangan performa antar kelas. Model sering kali menunjukkan *overfitting* terhadap kelas mayoritas, sementara kelas dengan data sedikit mengalami *underfitting* karena kurangnya contoh untuk dipelajari.

3) **Eksperimen balancing yang tidak menghasilkan perubahan signifikan.**

Penulis telah mencoba beberapa pendekatan balancing seperti oversampling dan undersampling. Namun, kedua metode tersebut tidak memberikan hasil yang lebih baik:

A. Oversampling meningkatkan jumlah data latih sehingga waktu training menjadi lebih lama tanpa peningkatan akurasi yang berarti.
B. Undersampling mempercepat training, tetapi menurunkan performa model karena sebagian data hilang dan informasi menjadi terbatas.

4) Lamanya proses fine-tuning dan training model.

Saat melakukan fine-tuning pada model pretrained,

penulis membuka sejumlah lapisan (layers) untuk dilatih ulang agar model dapat menyesuaikan dengan karakteristik dataset. Namun, semakin banyak layer yang dibuka, semakin berat pula komputasi yang dibutuhkan.

Ketika penulis menambah jumlah epoch dari 10 menjadi 15 dan jumlah layer fine-tuning dari 5 menjadi 10, waktu eksekusi meningkat signifikan. Meskipun langkah ini ditujukan untuk meningkatkan akurasi, durasi pelatihan per model dapat mencapai lebih dari 3 jam.

5) **Masalah pada stabilitas hasil model.**

Beberapa kali, model menunjukkan *fluctuating accuracy* di mana akurasi training meningkat, namun akurasi validasi justru menurun (*overfitting*). Hal ini menunjukkan bahwa model terlalu menyesuaikan diri dengan data latih dan kehilangan kemampuan generalisasi terhadap data baru.

3.1.2 Solusi atas Kendala yang Ditemukan

Selama proses pelatihan model klasifikasi citra rumah adat, sejumlah kendala teknis teridentifikasi, terutama berkaitan dengan ketidakseimbangan data (data imbalance), keterbatasan variasi citra, serta tantangan dalam menjaga keseimbangan antara akurasi dan generalisasi model. Untuk mengatasi berbagai kendala tersebut, beberapa strategi diterapkan secara sistematis, meliputi penerapan class weighting, peningkatan jumlah epoch dan fine-tuning, penggunaan early stopping, serta augmentasi data guna meningkatkan performa model secara menyeluruh.

```
# Compute class weights for balancing
class_weights = compute_class_weight(
    'balanced',
    classes=np.unique(train_df['label']),
    y=train_df['label']
)
class_weight_dict = dict(zip(np.unique(train_df['label']), class_weights))
print("\nClass weights:")
print(class_weight_dict)
```

Gambar 3.44 Melakukan Balancing Data

Kendala utama yang ditemukan adalah ketidakseimbangan distribusi kelas dalam dataset. Beberapa kelas rumah adat memiliki jumlah sampel jauh lebih sedikit dibandingkan kelas lainnya, sehingga model cenderung memberikan bias terhadap kelas mayoritas. Kondisi ini berpotensi menurunkan nilai recall dan precision pada kelas minoritas. Untuk mengatasi hal tersebut, diterapkan pendekatan class weighting, yaitu pemberian bobot lebih tinggi pada kelas dengan jumlah sampel lebih sedikit. Bobot kelas dihitung menggunakan fungsi compute_class_weight() dari pustaka *scikit-learn* dan diimplementasikan pada parameter class_weight dalam proses pelatihan model (model.fit()). Dengan pendekatan ini, kontribusi loss dari kelas minoritas menjadi lebih besar, sehingga model terdorong untuk belajar secara proporsional terhadap seluruh kelas. Hasil implementasi menunjukkan adanya peningkatan keseimbangan performa antar kelas, terutama dalam nilai *recall* dan *F1-score*.

```
IMG_SIZE = 224
BATCH_SIZE = 32
SEED = 42
INITIAL_EPOCHS = 15
FINE_TUNE_EPOCHS = 10
LR = 1e-3
```

Gambar 3.45 Epoch dan Fine Tuning

Selain itu, model awal yang dilatih selama 10 epoch menunjukkan hasil yang belum stabil, di mana nilai akurasi dan loss pada data validasi masih berfluktuasi. Untuk memperbaiki hal tersebut, dilakukan peningkatan jumlah epoch menjadi 15 agar model memiliki waktu belajar yang lebih optimal. Di sisi lain, proses fine-tuning juga diperluas, dari semula melibatkan 5 layer terakhir menjadi 10 layer terakhir dari model pra-latih (*pre-trained model*). Langkah ini memungkinkan model untuk menyesuaikan bobot parameter pada lapisan yang lebih dalam terhadap

karakteristik unik dari dataset rumah adat yang digunakan. Kombinasi antara peningkatan epoch dan fine-tuning terbukti efektif dalam meningkatkan akurasi validasi serta kemampuan model mengenali pola yang lebih kompleks pada citra.

```
checkpoint = ModelCheckpoint("xception_best.h5", save_best_only=True, monitor="val_accuracy", mode="max")
earlystop = EarlyStopping(monitor="val_accuracy", patience=3, restore_best_weights=True)

history = model.fit(
    train_generator,
    epochs=INITIAL_EPOCHS,
    validation_data=val_generator,
    callbacks=[checkpoint, earlystop],
    class_weight=class_weight_dict # Use class weights for balancing
)

# Fine-tuning: unfreeze top layers
for layer in model.layers[-30:]:  # unfreeze the last 30 layers
    if not isinstance(layer, tf.keras.layers.BatchNormalization):
        layer.trainable = True

model.compile(optimizer=Adam(learning_rate=1e-5),
              loss="categorical_crossentropy",
              metrics=["accuracy"])

history_ft = model.fit(
    train_generator,
    epochs=FINE_TUNE_EPOCHS,
    validation_data=val_generator,
    callbacks=[earlystop],
    class_weight=class_weight_dict # Use class weights for balancing
)
```

Gambar 3.46 Melakukan Earlystoping

Namun, peningkatan jumlah epoch juga menimbulkan risiko overfitting, di mana model terlalu menyesuaikan diri dengan data latih dan kehilangan kemampuan generalisasi terhadap data baru. Untuk mengatasi permasalahan ini, diterapkan metode Early Stopping yang berfungsi menghentikan pelatihan secara otomatis apabila tidak terjadi peningkatan performa validasi dalam beberapa iterasi berturut-turut. Pada penelitian ini, *callback* EarlyStopping dikonfigurasikan dengan parameter monitor='val_loss', patience=3, serta restore_best_weights=True. Penggunaan teknik ini mampu mengontrol proses pelatihan secara adaptif, menghindari pemborosan sumber daya komputasi, dan memastikan bobot terbaik tersimpan sebelum terjadi penurunan performa akibat overfitting.

```
# Data Augmentation and Preprocessing
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
)

val_datagen = ImageDataGenerator(rescale=1./255) # No augmentation for validation
```

Gambar 3.47 Data Augmentation

Selain pendekatan tersebut, dilakukan pula augmentasi data sebagai strategi tambahan untuk meningkatkan variasi dan kualitas data pelatihan. Proses augmentasi dilakukan menggunakan *ImageDataGenerator* dari TensorFlow Keras, dengan serangkaian transformasi seperti rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, serta fill_mode='nearest'. Setiap gambar juga dinormalisasi menggunakan parameter rescale=1./255. Teknik ini secara efektif memperluas keragaman dataset tanpa perlu menambah jumlah data secara manual, sehingga membantu model untuk beradaptasi terhadap berbagai variasi citra seperti perubahan orientasi, pencahayaan, dan posisi objek. Sementara itu, data validasi hanya dikenai proses rescaling agar evaluasi model tetap objektif tanpa pengaruh transformasi tambahan.

Secara keseluruhan, penerapan keempat solusi tersebut yaitu class weighting, peningkatan epoch dan fine-tuning, early stopping, dan augmentasi data terbukti memberikan hasil yang signifikan dalam meningkatkan performa dan stabilitas model. Model akhir menunjukkan kemampuan generalisasi yang lebih baik, penurunan bias terhadap kelas mayoritas, serta

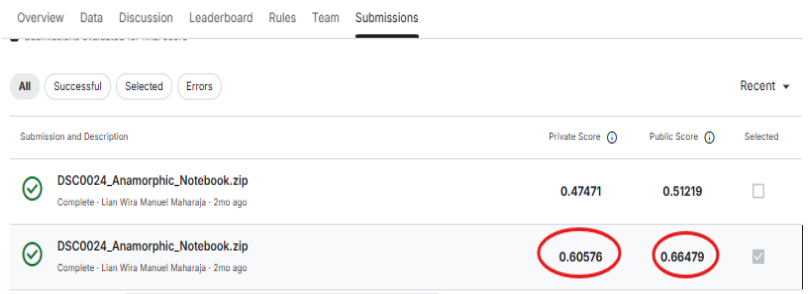
peningkatan akurasi dan konsistensi hasil prediksi pada seluruh kelas rumah adat. Pendekatan komprehensif ini juga memperlihatkan bahwa kombinasi strategi teknis dapat menjadi solusi efektif dalam menghadapi kendala umum pada proyek klasifikasi citra dengan distribusi data yang tidak seimbang.

3.4 Hasil Lomba/Kompetisi

3.4.1 Peringkat dan Score Kompetisi

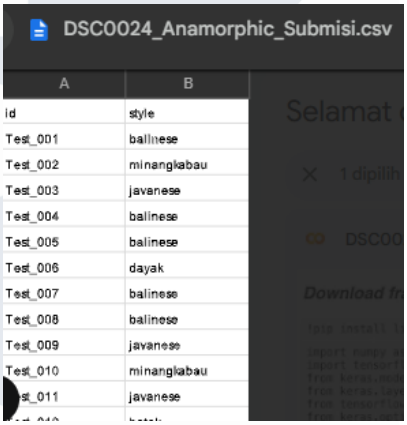
Setelah melalui seluruh tahapan penelitian yang meliputi pengumpulan data (data collecting), pra-pemrosesan data (data pre-processing), pemodelan (data modeling), pelatihan model (training), hingga evaluasi model, maka proses pengembangan model klasifikasi citra rumah adat di Indonesia telah selesai 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 3.48 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 3.48 Hasil Kompetisi

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 3.49.



Gambar 3.49 Hasil CSV

3.4.2 Pengembangan Mobilenetv2

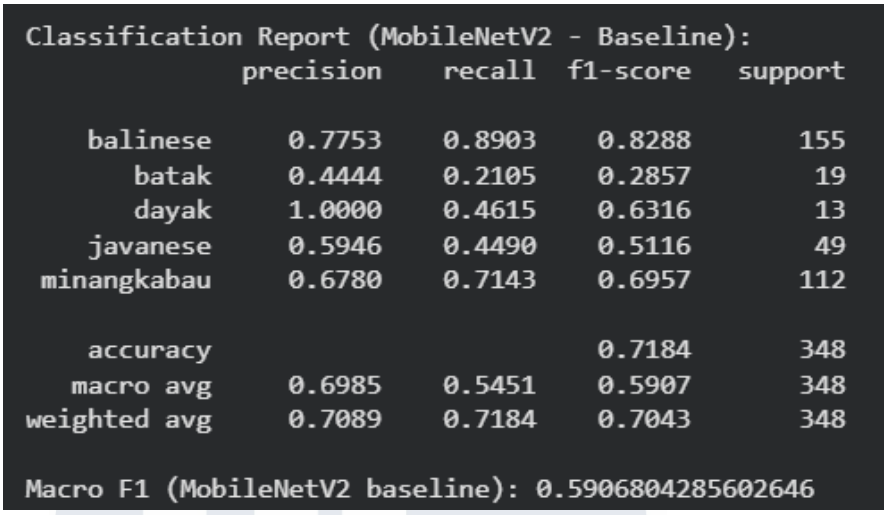
MobileNetV2 merupakan salah satu arsitektur Convolutional Neural Network yang digunakan dalam tahap awal pengembangan model klasifikasi citra rumah adat Nusantara. Pada tahap kompetisi LOGIKA UI 2025, MobileNetV2 digunakan sebagai salah satu model baseline dan hasil yang dikumpulkan (submission) merupakan model dengan performa validasi sebesar 66 persen dan macro F1-score sebesar 0,53. Model tersebut dikumpulkan sesuai dengan batas waktu kompetisi dan

menjadi capaian resmi tim dalam leaderboard.

Setelah periode submission berakhir, penulis melakukan pengembangan lanjutan terhadap MobileNetV2 dengan tujuan untuk mengevaluasi sejauh mana performa arsitektur ini masih dapat ditingkatkan apabila diberikan strategi pelatihan yang lebih optimal. Tahap pengembangan ini tidak dimaksudkan untuk menggantikan hasil submission, melainkan sebagai analisis eksploratif untuk memahami potensi maksimum MobileNetV2 sebagai single CNN architecture.

Pada tahap awal sebelum dilakukan pengembangan, model klasifikasi citra budaya Nusantara dibangun menggunakan arsitektur MobileNetV2 berbasis transfer learning. Model ini memanfaatkan bobot pralatih ImageNet dengan sebagian besar layer dibekukan pada tahap awal pelatihan, sehingga proses pembelajaran lebih difokuskan pada classifier di bagian akhir jaringan. Pendekatan ini memungkinkan model belajar lebih cepat dan mengurangi risiko overfitting, namun pada sisi lain membatasi kemampuan model dalam menyesuaikan diri dengan karakteristik visual spesifik citra budaya Nusantara.

Proses fine-tuning dilakukan dengan membuka sebagian layer terakhir MobileNetV2 dan menggunakan learning rate yang kecil. Meskipun demikian, arsitektur classifier yang relatif sederhana serta ketergantungan penuh pada fitur CNN menyebabkan model masih mengalami kesulitan dalam mengenali kelas budaya dengan jumlah data yang lebih sedikit. Hal ini tercermin dari hasil evaluasi, di mana akurasi model mencapai sekitar 66%, namun nilai macro F1-score masih rendah, yang mengindikasikan performa model belum merata pada seluruh kelas citra budaya Nusantara. Model cenderung lebih optimal pada kelas mayoritas, sementara kelas minoritas memiliki precision dan recall yang lebih rendah.



```
Classification Report (MobileNetV2 - Baseline):
              precision    recall  f1-score   support

    balinese     0.7753    0.8903    0.8288       155
       batak     0.4444    0.2105    0.2857        19
       dayak     1.0000    0.4615    0.6316        13
    javanese     0.5946    0.4490    0.5116        49
 minangkabau     0.6780    0.7143    0.6957       112

    accuracy                         0.7184       348
   macro avg     0.6985    0.5451    0.5907       348
weighted avg     0.7089    0.7184    0.7043       348

Macro F1 (MobileNetV2 baseline): 0.5906804285602646
```

Gambar 3.50 MobileNetv2 Submision

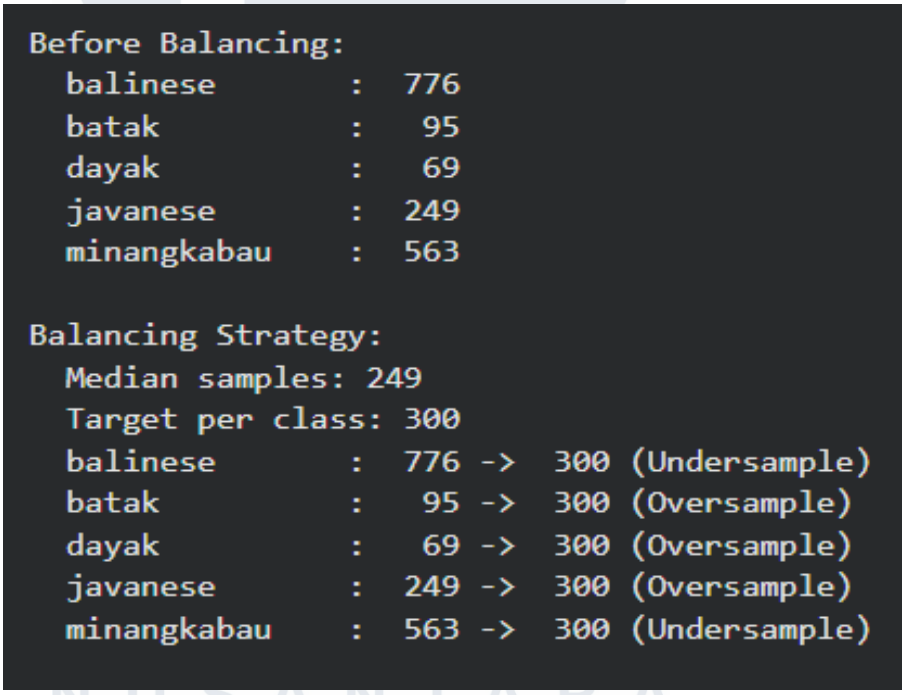
Gambar 3.50 menampilkan hasil evaluasi MobileNetV2 yang digunakan dan dikumpulkan pada tahap kompetisi DSC LOGIKA UI 2025. Model ini dilatih dengan konfigurasi dasar dan keterbatasan waktu eksperimen karena adanya deadline submission. Berdasarkan hasil evaluasi pada data validasi, model MobileNetV2 ini mencapai akurasi sebesar 66 persen dengan macro F1-score sebesar 0,5301.

Hasil tersebut menunjukkan bahwa secara umum model telah mampu mengenali pola visual rumah adat dengan cukup baik, khususnya pada kelas dengan jumlah data relatif besar dan ciri visual yang kuat seperti Balinese dan Minangkabau. Hal ini tercermin dari nilai F1-score kelas Balinese sebesar 0,81 dan Minangkabau sebesar 0,65. Namun, performa model menurun secara signifikan pada kelas minoritas seperti Batak dan Dayak, yang masing-masing hanya mencapai F1-score sebesar 0,25 dan 0,42.

Perbedaan yang cukup besar antara nilai accuracy (0,66) dan macro F1-score (0,53) mengindikasikan bahwa model masih mengalami ketidakseimbangan performa antar kelas. Model cenderung lebih akurat pada kelas mayoritas, tetapi kurang mampu melakukan generalisasi yang baik pada kelas dengan jumlah data lebih sedikit atau karakteristik visual

yang lebih kompleks. Kondisi ini menjadi indikasi awal adanya bias model terhadap kelas tertentu.

Tahap pengembangan dilakukan untuk mengevaluasi sejauh mana performa MobileNetV2 dapat ditingkatkan apabila tidak dibatasi oleh waktu submission. Pada tahap ini, MobileNetV2 tidak lagi digunakan sebagai end-to-end classifier, melainkan sebagai feature extractor, yang dikombinasikan dengan fitur Histogram of Oriented Gradients (HOG). Pendekatan ini memungkinkan model memanfaatkan keunggulan CNN dalam menangkap informasi visual tingkat tinggi sekaligus memanfaatkan HOG dalam menangkap pola tekstur dan struktur geometris bangunan rumah adat.



Gambar 3.51 Balancing MobileNetv2

Selain itu, dilakukan penyeimbangan data dengan oversampling pada kelas minoritas sehingga setiap kelas memiliki jumlah sampel yang relatif seimbang. Strategi ini bertujuan untuk mengurangi bias model terhadap kelas tertentu dan meningkatkan performa klasifikasi secara merata. Fitur hasil ekstraksi kemudian dinormalisasi dan digabungkan

menjadi satu vektor fitur yang lebih kaya sebelum diproses oleh jaringan klasifikasi.

Arsitektur klasifikasi pada tahap ini dirancang lebih dalam dengan beberapa lapisan dense, batch normalization, dan dropout bertahap. Pendekatan ini meningkatkan kapasitas model dalam mempelajari hubungan kompleks antar fitur sekaligus menjaga stabilitas proses pelatihan. Untuk meningkatkan generalisasi, digunakan teknik ensemble learning dengan beberapa learning rate berbeda, di mana hasil prediksi dari masing-masing model digabungkan melalui rata-rata probabilitas.

Classification Report:

	precision	recall	f1-score	support
balinese	0.7333	0.9778	0.8381	45
batak	0.9512	0.8667	0.9070	45
dayak	1.0000	1.0000	1.0000	45
javanese	0.9429	0.7333	0.8250	45
minangkabau	0.8864	0.8667	0.8764	45
accuracy			0.8889	225
macro avg	0.9028	0.8889	0.8893	225
weighted avg	0.9028	0.8889	0.8893	225

Gambar 3.52 Hasil Pengembangan MobileNetv2

Gambar 3.52 menunjukkan hasil evaluasi MobileNetV2 setelah dilakukan tahap pengembangan lanjutan pasca kompetisi. Pengembangan ini dilakukan dengan tujuan untuk mengevaluasi sejauh mana performa MobileNetV2 masih dapat ditingkatkan apabila diberikan strategi pelatihan yang lebih optimal, tanpa terikat oleh batasan waktu submission.

Berdasarkan hasil evaluasi, MobileNetV2 hasil pengembangan menunjukkan peningkatan performa yang sangat signifikan. Model mencapai akurasi sebesar 88,89 persen dengan macro F1-score sebesar 0,8893. Nilai ini menunjukkan bahwa model tidak hanya mengalami

peningkatan akurasi secara keseluruhan, tetapi juga mampu memberikan performa yang lebih seimbang pada seluruh kelas.

Peningkatan performa paling menonjol terlihat pada kelas Batak dan Dayak. Pada tahap submission, kedua kelas tersebut mengalami kesulitan klasifikasi dengan F1-score yang relatif rendah. Namun, setelah pengembangan, kelas Batak mencapai F1-score sebesar 0,9070 dan kelas Dayak mencapai F1-score sempurna sebesar 1,0000. Hal ini menunjukkan bahwa model telah mampu mempelajari ciri visual khas dari masing-masing kelas secara lebih representatif.

Selain itu, kelas Javanese dan Minangkabau juga mengalami peningkatan yang signifikan, masing-masing mencapai F1-score sebesar 0,8250 dan 0,8764. Peningkatan ini menunjukkan bahwa model memiliki kemampuan generalisasi yang lebih baik dalam menghadapi variasi bentuk dan struktur rumah adat pada masing-masing kelas.