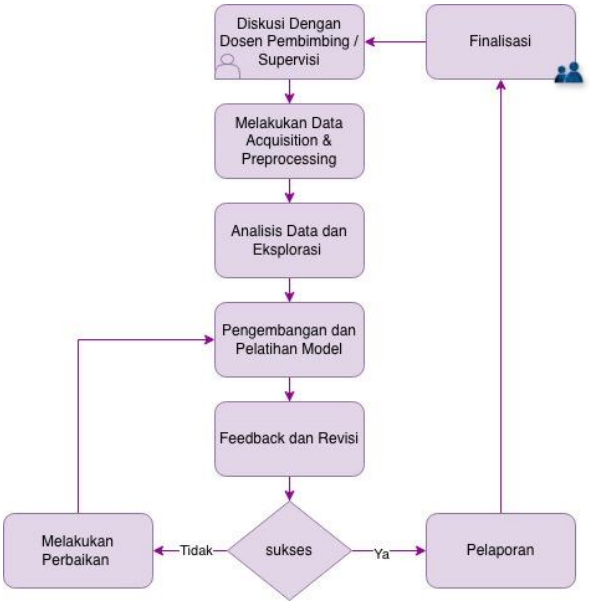
BAB III

PELAKSANAAN PRO-STEP : ROAD TO CHAMPION

3.1 Kedudukan dan Koordinasi

Dalam pelaksanaan PRO-STEP: Road to Champion Program, berpartisipasi sebagai anggota tim dalam Data Science Competition (DSC) LOGIKA UI 2025, yang merupakan salah satu rangkaian kegiatan dari LOGIKA UI (Lomba Karya dan Inovasi Mahasiswa Komputer Universitas Indonesia). Pada kompetisi ini, berperan sebagai Data Scientist sekaligus menjadi penanggung jawab utama dalam pengembangan model serta analisis hasil prediksi menggunakan metode machine learning dan deep learning.

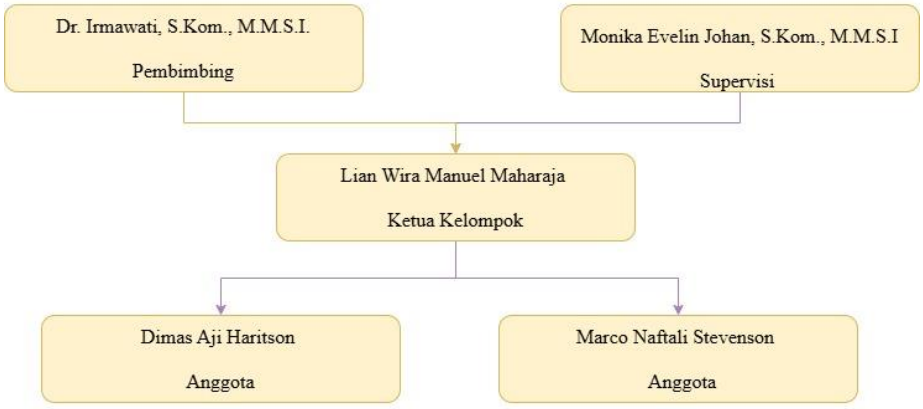


Gambar 3.1 Alur Koordinasi

Secara struktural, kegiatan ini dilaksanakan di bawah bimbingan Dr. Irmawati, S.Kom., M.M.S.I. selaku Dosen Pembimbing Internal dari Program Studi Sistem Informasi, Universitas Multimedia Nusantara (UMN). Beliau berperan memberikan arahan akademik, bimbingan metodologis, serta memastikan pelaksanaan proyek

sesuai dengan prinsip ilmiah, etika penelitian, dan capaian pembelajaran yang telah ditetapkan.

Selain itu, proyek ini juga berada dalam pengawasan Monika Evelin Johan, S.Kom., M.M.S.I. selaku Supervisor Lapangan, yang memberikan panduan teknis, melakukan evaluasi terhadap perkembangan tim, serta memastikan kesesuaian hasil kerja dengan target kompetisi dan standar profesional di bidang data science. Koordinasi antara pihak pembimbing dan tim dilakukan secara berjenjang serta kolaboratif. Dosen pembimbing internal memberikan arahan konseptual dan metodologis, sedangkan supervisor lapangan memantau pelaksanaan teknis dan pencapaian hasil. Setiap anggota tim juga rutin melakukan koordinasi melalui rapat daring dan grup komunikasi internal.



Gambar 3.2 Struktur Kedudukan

Struktur Kedudukan pada gambar 3.1 berikut dijelaskan bahwa:

1. Dosen Pembimbing Internal: Dr. Irmawati, S.Kom., M.M.S.I

Memberikan bimbingan akademik, evaluasi, serta memastikan keaslian dan relevansi proyek dengan bidang ilmu yang digeluti dan melakukan supervisi berkala terhadap laporan kemajuan dan hasil akhir proyek.

2. Supervisor Lapangan: Monika Evelin Johan, S.Kom., M.M.S.I.

Memberikan arahan teknis terkait pengembangan model machine learning dan pengelolaan data dan mengevaluasi efisiensi implementasi serta kinerja model dalam konteks kompetisi.

Serta pada gambar 3.2 Peran anggota Tim Kompetisi DSC LOGIKA UI 2025

yaitu;

1. Lian Wira Manuel Maharaja; (Data Engineer)

Bertanggung jawab dalam proses data acquisition, data preprocessing, serta membangun data pipeline yang efisien, terstruktur dan perbandingan kinerja tiga model berbeda MobileNetV2, ResNet50V2, DenseNet121, VGG16 untuk menentukan model dengan performa terbaik

2. Marco Naftali Stevenson ; Data Analyst

Melaksanakan Exploratory Data Analysis (EDA), menyusun visualisasi data, serta mengembangkan dashboard untuk penyajian insight hasil analisis.

3. Dimas Aji Haritson ; Data Scientist

Mendesain dan mengimplementasikan model deep learning berbasis Convolutional Neural Network (CNN) untuk klasifikasi citra rumah adat NusantaraDenganPendekatan Hybrid Feature Extraction dan Ensemble Learning

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

Berikut pada Tabel 3.1 Detail Pekerjaan yang dilakukan dalam PRO-STEP yang merupakan rangkuman pengerjaan mingguan selama mengikuti kompetisi yang dimulai dari awal kompetisi hingga pengerjaan kompetisi.

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

No	Minggu	Proyek	Keterangan
1	1	Penerimaan Brief Lomba	Menerima dan mempelajari secara menyeluruh brief kompetisi LOGIKA UI, termasuk alur pelaksanaan lomba, ruang lingkup permasalahan, serta kriteria dan mekanisme penilaian yang ditetapkan oleh penyelenggara.
2	2	Diskusi Awal dan Brainstorming dengan Pembimbing	Melaksanakan diskusi awal bersama dosen pembimbing untuk memahami tujuan kegiatan PRO-STEP, ruang lingkup lomba, serta menentukan pendekatan dan strategi pengerjaan proyek.

3	3	Penentuan Tema dan Pembagian Tugas	Menetapkan tema lomba *Peran Ilmu Matematika dalam Mewujudkan Kota Cerdas* serta melakukan pembagian tugas dan tanggung jawab anggota tim berdasarkan kompetensi masing-masing.
4	4	Perancangan Data Pipeline Tahap Awal	Melakukan perancangan awal *data pipeline* menggunakan ImageDataGenerator sebagai dasar pengolahan data citra, serta melakukan koordinasi teknis dengan supervisor.
5	5	Pengembangan dan Penyempurnaan Data Pipeline	Menyempurnakan struktur dan alur *data pipeline* agar lebih stabil dan optimal sebagai persiapan untuk proses eksplorasi dan pelatihan model.
6	6	Eksplorasi Pipeline dan Model Awal	Melakukan eksplorasi awal terhadap pipeline dan model klasifikasi visual, serta mengevaluasi performa awal menggunakan metrik F1-Score sebagai dasar validasi.
7	7	Eksplorasi dan Iterasi Model Lanjutan	Melaksanakan percobaan berulang dan penyesuaian konfigurasi model untuk meningkatkan performa, akurasi, serta kestabilan hasil prediksi.
8	8	Penyusunan Bab I Laporan	Menyusun Bab I laporan PRO-STEP yang mencakup latar belakang, rumusan masalah, tujuan kegiatan, manfaat penelitian, serta gambaran umum timeline pelaksanaan.
9	9	Eksperimen Model dan Evaluasi Mendalam	Melakukan eksperimen lanjutan dengan berbagai skenario parameter, mengevaluasi performa model secara komprehensif, serta melaporkan progres kepada supervisor.
10	10	Finalisasi Model dan Submission	Melakukan finalisasi model terbaik berdasarkan hasil evaluasi, serta melakukan submission file IPYNB dan CSV ke platform Kaggle sesuai dengan ketentuan babak penyisihan.

11	11	Bimbingan dan Finalisasi Bab I dan Persiapan Bab III	Melaksanakan bimbingan lanjutan untuk penyempurnaan Bab I serta mempersiapkan struktur dan kebutuhan penulisan Bab III (Metodologi Penelitian).
12	12	Penyusunan Bab III – Metodologi Penelitian	Menyusun Bab III laporan yang mencakup desain penelitian, alur sistem, metode pengolahan data, arsitektur model, serta skema evaluasi yang digunakan.
13	13	Implementasi Metodologi dan Dokumentasi	Mengimplementasikan metode yang telah dirancang pada Bab III serta mendokumentasikan proses eksperimen dan hasil implementasi secara sistematis.
14	14	Penyusunan Bab III – Hasil dan Pembahasan	Menyusun Bab III laporan yang berisi hasil eksperimen model, analisis performa, perbandingan metode, serta pembahasan terhadap temuan yang diperoleh.
15	15	Analisis dan Interpretasi Hasil	Melakukan analisis lanjutan terhadap hasil eksperimen, interpretasi performa model, serta mengaitkan hasil penelitian dengan tujuan dan teori yang relevan.
16	16	Penyusunan Bab IV – Kesimpulan dan Saran	Menyusun Bab IV laporan yang mencakup kesimpulan akhir penelitian serta saran untuk pengembangan sistem dan penelitian selanjutnya.
17	17	Finalisasi Laporan Akhir PRO-STEP	Melakukan penyelarasan seluruh bab laporan, pengecekan konsistensi penulisan, serta finalisasi dokumen laporan PRO-STEP secara keseluruhan.

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

Selama kegiatan PRO-STEP jalur lomba, mengikuti serangkaian proses kompetisi yang terdiri dari perencanaan, pengembangan model, evaluasi, hingga penyusunan laporan..

3.3.1 Proses Pelaksanaan

Aktivitas dikerjakan melalui koordinasi tim, bimbingan pembimbing, serta supervisi teknis secara berkala sebagai berikut;

3.3.1.1 Tahap 1 Perencanaan dan Penentuan Arah Proyek

Pada tahap awal ini, dan tim memulai proses dengan melakukan analisis komprehensif terhadap *brief* lomba yang diberikan oleh panitia Data Science Competition (DSC) LOGIKA UI 2025. Analisis dilakukan untuk memahami batasan kompetisi, aturan penggunaan model, ketentuan submisi, struktur penilaian, serta ruang lingkup permasalahan yang harus diselesaikan. Salah satu poin penting yang diperoleh dari tahap ini adalah larangan penggunaan model berbasis API dan AutoML, sehingga seluruh proses pemodelan harus dirancang secara mandiri menggunakan *framework* machine learning, khususnya TensorFlow–Keras.

Setelah memahami *brief*, melakukan diskusi awal bersama dosen pembimbing (Dr. Irmawati, S.Kom., M.M.S.I) untuk memastikan arah proyek sesuai dengan standar akademik dan etika kompetisi. Diskusi meliputi penentuan metodologi yang akan digunakan, strategi teknis, serta pendekatan analitis dalam pengembangan pipeline dan model. Pada tahap ini pula dilakukan penentuan tema yang akan diangkat, yaitu “Optimasi Model Transfer Learning untuk Klasifikasi Gambar Rumah Adat Indonesia”, sesuai topik resmi kompetisi.

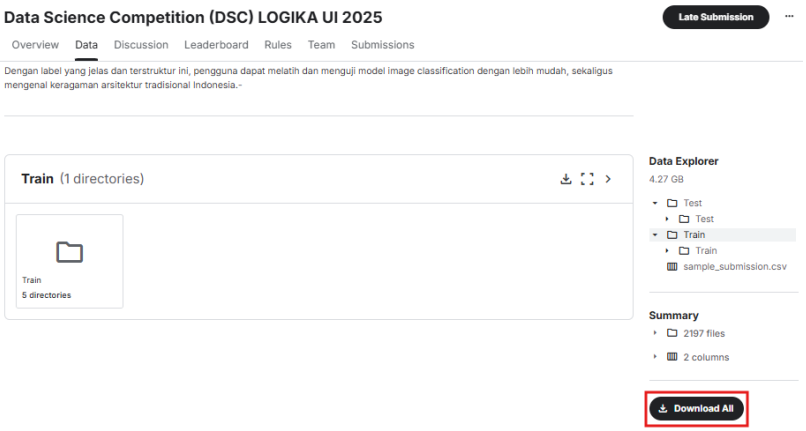
Berikutnya, tim melakukan proses brainstorming untuk menetapkan pembagian tugas (jobdesk) yang jelas sesuai kemampuan masing-masing anggota. bertanggung jawab pada rancangan pipeline data dan pengolaan dataset, sementara anggota lain fokus pada EDA dan perancangan model. Pembagian peran yang terstruktur ini membantu mempercepat alur kerja sekaligus menjaga

konsistensi hasil. Selain itu, melakukan pencarian referensi metodologis untuk memperkuat landasan proyek. Referensi diperoleh melalui jurnal ilmiah, artikel *peer-reviewed*, dokumentasi Kaggle, dan portofolio karya kompetisi sejenis. Fokus kajian diarahkan pada:

1. Teknik *data augmentation* pada dataset citra budaya,
2. Permasalahan *class imbalance* pada citra bangunan,
3. Studi sebelumnya mengenai efisiensi arsitektur transfer learning seperti MobileNetV2, ResNet50V2, DenseNet121, VGG16.
4. Tren penilaian kompetisi data science yang menekankan reproduktibilitas, struktur notebook, dan *explainability*.

3.3.1.2 Tahap 2 Perancangan Data Pipeline

Membangun pipeline data menggunakan *ImageDataGenerator* untuk proses augmentasi, balancing, dan standarisasi citra sebagai input model. Tahap ini penting untuk meningkatkan kualitas dataset serta mencegah overfitting. Sebelum membangun *pipeline* data menggunakan *ImageDataGenerator*, terlebih dahulu melakukan proses identifikasi dan inspeksi awal terhadap struktur dataset. Tahap ini diawali dengan mengunggah berkas dataset yang telah diunduh dari platform Kaggle ke dalam Google Drive agar dapat diproses melalui lingkungan Google Colab;



Gambar 3.3 Dataset Lomba DSC Logika UI 2025

Sumber : Kaggle.com

Dataset tersebut kemudian dipetakan ke dalam direktori *TRAIN_DIR* dan *TEST_DIR* untuk memastikan seluruh berkas citra dapat diakses dengan benar oleh sistem.

```
from google.colab import drive
drive.mount('/content/drive')
TRAIN_DIR = os.path.join("/content/drive/MyDrive/dsc-logika-ui-2025/Train/Train")
TEST_DIR = os.path.join("/content/drive/MyDrive/dsc-logika-ui-2025/Test/Test")
SAMPLE_SUB_PATH = os.path.join("/content/drive/MyDrive/dsc-logika-ui-2025/sample_submission.csv")

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

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

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

Gambar 3.4 Code membaca struktur folder pada direktori *TRAIN_DIR*

Setelah dataset berhasil dimuat, melakukan pemeriksaan jumlah kategori (kelas) dengan membaca nama-nama folder yang terdapat pada direktori *Train*. Tahap ini penting untuk memverifikasi bahwa struktur dataset telah sesuai format klasifikasi citra berbasis CNN yang terdapat pada data set dibawah ini dengan code berikut ini ;

```
categories = os.listdir(TRAIN_DIR)

# Set up subplots
num_categories = len(categories)
num_images_per_category = 5
fig, axes = plt.subplots(num_categories, num_images_per_category, figsize=(5, 5))

for i, category in enumerate(categories):
    category_path = os.path.join(TRAIN_DIR, category)
    image_names = os.listdir(category_path)[:num_images_per_category]  # Get the first 5 images in the category

    for j, image_name in enumerate(image_names):
        image_path = os.path.join(category_path, image_name)
        img = plt.imread(image_path)

        # Plot images in subplots
        axes[i, j].imshow(img)
        axes[i, j].set_title(category)
        axes[i, j].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```

Gambar 3.5 Code menampilkan Visualisasi masing-masing kelas

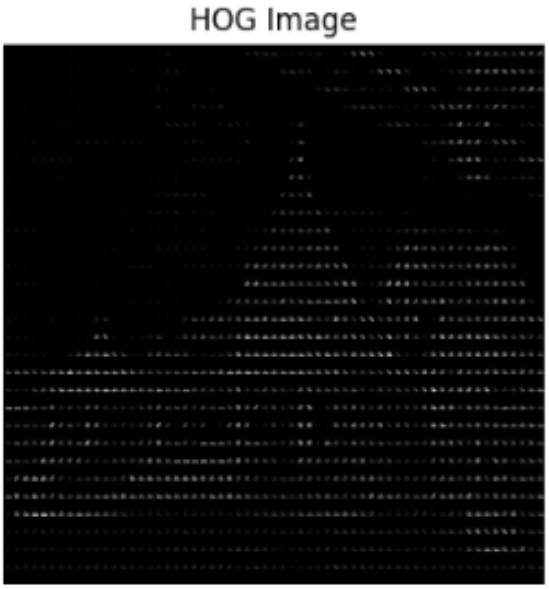
Gambar 3.6 Visualisasi Sampel Gambar Masing Masing Kelas

Setiap baris pada gambar mewakili satu kategori rumah adat. Dari visualisasi terlihat bahwa:

1. Citra pada kategori Balinese memiliki arsitektur khas gapura dan pura.
2. Kategori Batak menunjukkan rumah adat berbentuk segitiga dengan atap runcing.
3. Kategori Javanese didominasi rumah joglo dengan atap lebar.
4. Kategori Dayak banyak menampilkan rumah panjang (long house) dan ornamen kayu khas Kalimantan.
5. Kategori Minangkabau terlihat jelas bentuk atap gonjong yang melengkung ke atas.

Sebelum melakukan pelatihan model berbasis deep learning, terlebih dahulu melakukan tahap *baseline modeling* untuk mengetahui karakteristik fitur citra dan melihat sejauh mana model klasik dapat mempelajari pola visual pada dataset rumah adat Indonesia. Proses ini penting dilakukan untuk memahami tingkat kesulitan dataset, mendeteksi potensi ketimpangan kelas, serta menjadi acuan dalam menentukan arah tuning parameter pada model deep learning berikutnya; menerapkan metode *feature extraction* berbasis Histogram of Oriented Gradients (HOG) untuk mengekstraksi fitur tepi dan pola tekstur dari setiap citra. HOG bekerja dengan menghitung distribusi orientasi gradien dalam sel-sel kecil pada

gambar



Gambar 3.7 Visualisasi sampel gambar HOG

Metode Dua fungsi dibuat untuk mendukung proses ini:

```python
# ✅ Khusus training
def extract_features_train(image_path):
    img = cv2.imread(image_path)
    resized_img = cv2.resize(img, (224, 224))
    gray_img = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
    hog_features = hog(
        gray_img,
        orientations=9,
        pixels_per_cell=(8, 4),
        cells_per_block=(2, 2),
        transform_sqrt=True,
        visualize=False
    )
    return np.array(hog_features)
```

Gambar 3.8 Code traning sampel gambar HOG

Digunakan khusus untuk ekstraksi fitur pada seluruh gambar *train.* Tahapan prosesnya meliputi:

1. Membaca gambar menggunakan OpenCV.
2. *Resize* ke ukuran 224×224 piksel.
3. Konversi menjadi grayscale.
4. Menghasilkan vektor fitur HOG berdimensi 53.640 fitur per gambar.

Vektor inilah yang digunakan untuk training SVM

```python
# ✅ Khusus visualisasi
def extract_features_viz(image_path):
    img = cv2.imread(image_path)
    resized_img = cv2.resize(img, (224, 224))
    gray_img = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
    hog_features, hog_image = hog(
        gray_img,
        orientations=9,
        pixels_per_cell=(8, 4),
        cells_per_block=(2, 2),
        transform_sqrt=True,
        visualize=True
    )
    return hog_features, hog_image
```

Gambar 3.9 Code visualisasi sampel gambar HOG

Fungsi ini, selain menghasilkan vektor fitur, juga mengembalikan gambar representasi HOG (*HOG visualization image*) untuk melihat pola gradien yang terdeteksi. Selanjutnya pada tahap penyusunan Seluruh citra pada direktori TRAIN_DIR diproses menggunakan fungsi HOG sehingga menghasilkan:

X = array fitur (1401 × 53.640)

y = label kelas (dengan *encoding* 0–4)

Dataset kemudian dibagi menjadi *training set* dan *testing set* dengan proporsi 80:20. Hasil *split* menunjukkan dimensi:

```python
X = np.array(X)
y = np.array(y)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train.shape, X_test.shape
```

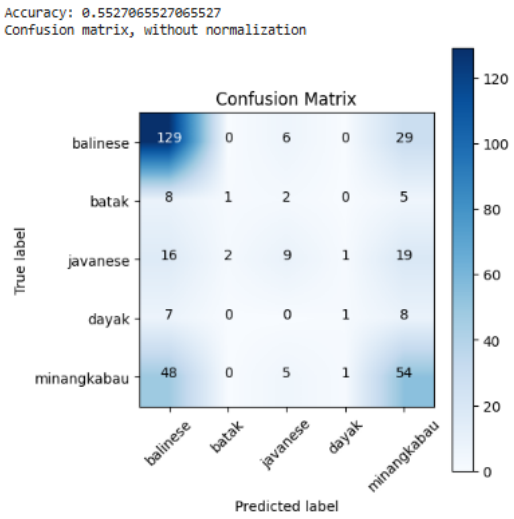
```
((1401, 53460), (351, 53460))
```

Gambar 3.10 Split dataset HOG

Hal ini menunjukkan bahwa setiap citra direpresentasikan dalam vektor fitur berukuran sangat besar yang juga menandakan metode tradisional seperti SVM mungkin menghadapi tantangan pada dataset bertekstur kompleks melatih dua pendekatan SVM yaitu;

1. SVM Standar (One-vs-One default)

Pada tahap ini menggunakan kernel linear dan Melatih model biner untuk setiap pasangan kelas

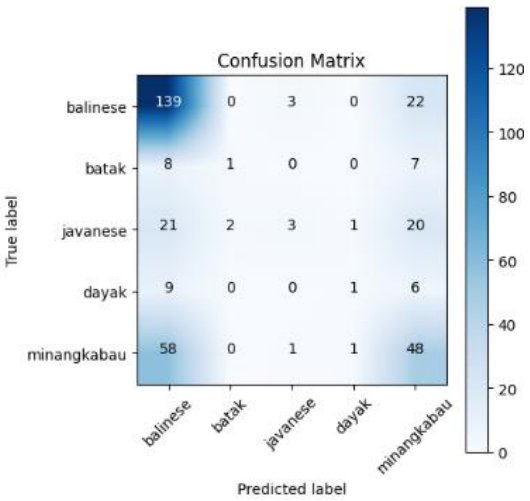


Gambar 3.11 Confusion matrix SVM Standar

Confusion Matrix pada gambar 3.11 menunjukkan bahwa beberapa kelas seperti *Balinese* dan *Minangkabau* cukup teridentifikasi, namun kelas berukuran kecil seperti *Batak* dan *Dayak* sulit dikenali, menunjukkan adanya *class imbalance* dan kompleksitas fitur visual yang tinggi

2. SVM dengan One-vs-Rest (OVR)

Pada tahap ini Melatih satu model untuk setiap kelas dan Setiap model membedakan “kelas saya” vs “bukan kelas saya” Hasil akurasi meningkat menjadi 0.547;



Gambar 3.12 Confusion matrix SVM dengan One-vs-Rest (OVR)

menyatakan untuk menilai apakah terdapat perbedaan performa yang

signifikan antara model SVM standar dan SVM dengan pendekatan One-vs-Rest (OVR), melakukan dua jenis uji statistik, yaitu McNemar Test dan Wilcoxon Signed-Rank Test. Hasil McNemar Test menunjukkan nilai *statistic* sebesar 0.0 dengan *p-value* 1.0. Karena nilai *p-value* lebih besar dari ambang signifikansi 0.05, maka dapat disimpulkan bahwa tidak terdapat perbedaan yang signifikan secara statistik antara distribusi kesalahan prediksi kedua model. Ini berarti bahwa dari sisi perbandingan kategori prediksi benar–salah, performa kedua model tidak menunjukkan perubahan berarti. Namun, hasil yang berbeda ditunjukkan oleh Wilcoxon Signed-Rank Test, yang dilakukan untuk membandingkan performa kedua model secara *pairwise* berdasarkan hasil prediksi individu. Uji ini menghasilkan *statistic* sebesar 579.0 dengan *p-value* 0.00314, yang berada di bawah ambang 0.05. Dengan demikian, dapat disimpulkan bahwa terdapat perbedaan performa yang signifikan secara statistik antara SVM dan OVR. Artinya, meskipun secara keseluruhan distribusi kesalahan kedua model tidak berbeda signifikan (berdasarkan McNemar), namun ketika dievaluasi secara pasangan per sampel, model OVR menunjukkan perubahan prediksi yang cukup konsisten untuk dianggap lebih baik dibandingkan SVM standar.

Tahap HOG + SVM digunakan sebagai baseline untuk memahami kemampuan model tradisional dalam mengenali rumah adat. Hasilnya menunjukkan bahwa akurasi masih rendah dan variabilitas antar kelas tinggi, sehingga pendekatan deep learning diperlukan sebagai model utama.

Pada tahap ini, beberapa parameter penting ditetapkan untuk memastikan proses pelatihan model berlangsung stabil, terkontrol, dan sesuai standar praktik terbaik dalam implementasi CNN berbasis transfer learning. Parameter dasar yang digunakan meliputi ukuran input gambar (IMG_SIZE dan IMG_SIZE) yang ditetapkan sebesar 224×224 piksel, sesuai dengan standar arsitektur pretrained seperti ResNet, EfficientNet, dan MobileNet [10]. Dimensi tersebut kemudian direpresentasikan dalam variabel *input_shape* dengan format (224, 224, 3) sebagai masukan tiga kanal RGB.

```
num_of_classes = 5
IMG_SIZE = 224
BATCH_SIZE = 32
SEED = 42
INITIAL_EPOCHS = 20
FINE_TUNE_EPOCHS = 15
DROPOUT_RATE = 0.4      # Dropout di head classifier
LEARNING_RATE = 1e-3     # Learning rate awal
```

Gambar 3.13 Parameter yang digunakan

Ukuran *batch* ditentukan sebesar 32, yang merupakan kompromi antara stabilitas pembaruan gradien dan efisiensi memori saat pelatihan. Jumlah kelas yang akan diprediksi ditetapkan sebanyak 5, sesuai dengan lima kategori rumah adat pada dataset. Parameter lain seperti *SEED* = 42 digunakan untuk memastikan reproduktibilitas hasil eksperimen. Proses pelatihan dibagi menjadi dua tahap, yaitu *INITIAL_EPOCHS* = 20 untuk melatih *head classifier* pada lapisan atas model, dan *FINE_TUNE_EPOCHS* = 15 untuk melakukan penyesuaian ulang (*fine-tuning*) pada sebagian lapisan model pretrained agar model mampu mempelajari fitur domain spesifik rumah adat secara lebih mendalam.

Selain itu, digunakan nilai *DROPOUT_RATE* = 0.4 pada bagian classifier untuk mencegah overfitting dengan cara mengacak neuron selama pelatihan. Sedangkan *learning rate awal* ditetapkan sebesar 1e-3 guna memastikan proses optimisasi berlangsung stabil ketika melatih lapisan baru sebelum memasuki tahap *fine-tuning*. Seluruh parameter ini dipilih berdasarkan eksperimen awal dan rekomendasi literatur sehingga model dapat belajar secara optimal pada dataset yang relatif beragam namun tidak terlalu besar.

Dataset rumah adat Indonesia memiliki variasi yang cukup besar dari sisi pencahayaan, sudut pandang kamera, serta latar belakang.

```
# Untuk kelas mayoritas (augmentasi ringan)
train_datagen_normal =  ImageDataGenerator(
    rescale=1./255,
    rotation_range=0.5,
    zoom_range=0.005,
    horizontal_flip=True,
)

# Untuk kelas minoritas (augmentasi sedikit lebih kuat, tapi masih natural)
train_datagen_strong = ImageDataGenerator(
    rescale=1./255,
    rotation_range=1.0,
    width_shift_range=0.01,
    height_shift_range=0.01,
    zoom_range=0.01,
    brightness_range=[0.97, 1.03],
    horizontal_flip=True,
)

# Validasi (tanpa augmentasi)
test_datagen = ImageDataGenerator(rescale=1./255)
```

Gambar 3.14 Gambar Data Augmentasi

Oleh karena itu, menerapkan *data augmentation* sebagai upaya memperkaya distribusi data dan mencegah model mengalami overfitting. Strategi augmentasi dibagi menjadi dua skenario:

1. Kelas Mayoritas (augmentasi ringan / subtle augmentation)

Kelas *Balinese, Javanese*, dan *Minangkabau* memiliki jumlah sampel relatif besar. Untuk mencegah distorsi berlebihan, augmentasi dibuat lebih halus:

1. rotasi ≤ 0.5°
2. zoom ≤ 0.5%
3. *horizontal flip*
4. normalisasi piksel 1./255

Augmentasi ringan ini memastikan citra berubah tetapi tidak mengaburkan pola arsitektur khas rumah adat.

2. Kelas minoritas (augmentasi lebih kuat namun tetap natural)

Kelas *Batak* dan *Dayak* memiliki jumlah data yang jauh lebih sedikit. Untuk memperkaya variasi data, ditambahkan augmentasi tambahan:

1. rotasi hingga 1°
2. pergeseran kecil (0.01)
3. zoom hingga 1%
4. variasi kecerahan ±3%
5. *horizontal flip*

Pendekatan ini membantu model mempelajari pola visual pada kelas minoritas dengan lebih baik.

3. Pembangunan Test Generator (Tanpa Augmentasi)

Untuk dataset uji, memastikan proses pembacaan citra berjalan konsisten tanpa transformasi apa pun selain *rescaling* Hal ini penting agar prediksi yang dihasilkan model tidak dipengaruhi modifikasi tambahan yang tidak relevan.

Setelah itu, dilakukan Pembangunan DataFrame Training dan Validation dengan Stratified Split. Pada tahap ini, membangun sebuah *DataFrame* yang berisi seluruh informasi gambar dan labelnya dari direktori TRAIN_DIR. Proses ini merupakan fondasi penting dari *data pipeline*, karena seluruh tahap augmentasi, balancing, dan pelatihan model bertumpu pada struktur data yang rapi dan terstandardisasi.

Setelah Proses ini merupakan fondasi penting dari *data pipeline*, karena seluruh tahap augmentasi, balancing, dan pelatihan model bertumpu pada struktur data yang rapi dan terstandardisasi. Langkah pertama adalah membaca seluruh folder kelas yang terdapat pada direktori TRAIN_DIR sebagai gambar berikut;

```
# ------------------ Build DataFrame ------------------
all_classes = sorted(os.listdir(TRAIN_DIR))
filepaths = []
labels = []

for cls in all_classes:
    cls_dir = os.path.join(TRAIN_DIR, cls)
    for f in os.listdir(cls_dir):
        if f.lower().endswith(('.jpg', '.png', '.jpeg')):
            filepaths.append(os.path.join(cls_dir, f))
            labels.append(cls)
```

Gambar 3.15 Code Membaca Seluruh Folder Kelas

Setiap nama folder mewakili satu kelas rumah adat: balinese, javanese, minangkabau, batak, dan dayak. Setelah seluruh citra berhasil dimasukkan ke dalam DataFrame, langkah selanjutnya adalah melakukan pemisahan dataset menjadi subset *training* dan *validation*. Namun, pemisahan tidak dilakukan secara acak, melainkan menggunakan pendekatan stratified split, seperti terlihat pada kode berikut:

```python
# ------------------ Split train/val ------------------
train_df, val_df = train_test_split(
    df,
    test_size=0.2,
    stratify=df['label'],
    random_state=SEED
)
print("Train size:", len(train_df))
print("Val size:", len(val_df))
```

Gambar 3.16 Code Split Data Train/Val

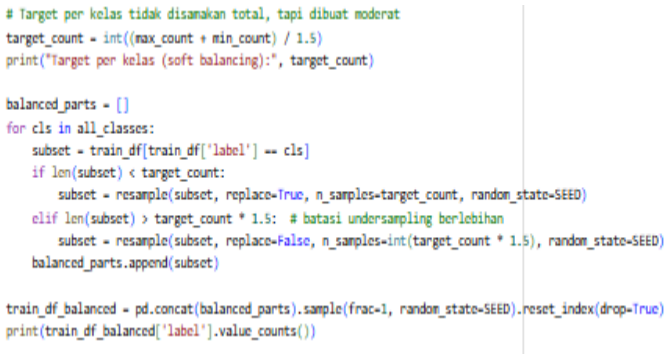
Pada dataset ini terdapat ketidakseimbangan jumlah sampel antar kelas (imbalanced data), seperti gambar berikut: kelas balinese memiliki jumlah gambar yang jauh lebih besar, sedangkan kelas batak dan dayak memiliki jumlah yang sedikit.

Jika pemisahan dilakukan tanpa stratifikasi, ada kemungkinan kelas minoritas tidak muncul dalam validation set, atau proporsinya berubah terlalu jauh dari distribusi awal. Hal tersebut dapat menyebabkan dua masalah utama:

```
Total images: 1752
label
balinese       776
minangkabau    563
javanese       249
batak           95
dayak           69
Name: count, dtype: int64
Train size: 1401
Val size: 351
```

Gambar 3.17 Hasil Split Data

Sebagai gambar 3.17 berikut pertama; evaluasi model menjadi tidak skurat jika validation set tidak mengandung perwakilan dari semua kelas, maka: model tidak diuji pada seluruh kelas, nilai akurasi dan metrik evaluasi lainnya menjadi bias, dan model seakan terlihat baik padahal tidak menguasai kelas kecil. Kedua; Model Tidak Belajar Secara Adil Stratified training memastikan proporsi setiap kelas dalam training tetap serupa seperti data aslinya. Misalnya jika kelas Dayak hanya memiliki 69 sampel, stratifikasi memastikan sekitar 80% tetap berada di training (±55 gambar) dan sisanya tetap masuk validation 20% (±14 gambar), bukan hilang sama sekali. Setelah itu dilakukan Class Balancing dengan Soft Balancing Pada tahap ini, tidak melakukan oversampling penuh maupun undersampling ekstrem seperti gambar code dibawah berikut;



```
# Target per kelas tidak disamakan total, tapi dibuat moderat
target_count = int((max_count + min_count) / 1.5)
print("Target per kelas (soft balancing):", target_count)

balanced_parts = []
for cls in all_classes:
    subset = train_df[train_df['label'] == cls]
    if len(subset) < target_count:
        subset = resample(subset, replace=True, n_samples=target_count, random_state=SEED)
    elif len(subset) > target_count * 1.5:  # batasi undersampling berlebihan
        subset = resample(subset, replace=False, n_samples=int(target_count * 1.5), random_state=SEED)
    balanced_parts.append(subset)

train_df_balanced = pd.concat(balanced_parts).sample(frac=1, random_state=SEED).reset_index(drop=True)
print(train_df_balanced['label'].value_counts())
```

Gambar 3.18 Code Class Balancing Data

Karena keduanya berisiko menimbulkan yaitu duplikasi sampel berlebihan (oversampling full) dan hilangnya karakteristik data penting (undersampling full). Sebagai gantinya digunakan metode soft balancing menghasilkan seperti gambar dibawah ini:

```
Target per kelas (soft balancing): 375
label
balinese       450
minangkabau    450
dayak          375
batak          375
javanese       375
Name: count, dtype: int64
Found 2025 validated image filenames belonging to 5 classes.
Found 351 validated image filenames belonging to 5 classes.
```

Gambar 3.19 Hasil Class Balancing Data (Soft Balancing)

Pada gambar 3.19 yaitu menentukan target_count moderat di antara kelas terbesar dan terkecil: kelas besar disamakan hingga ±450, kelas minoritas di-resample hingga 375 dan kelas mayoritas tidak dipotong berlebihan (maks 150% target). Hasil akhirnya distribusi data menjadi lebih merata tanpa menghilangkan informasi asli dataset.

Setelah dilakukan balancing data maka tahap selanjutnya membuat Data Generator Training dan Validation. Generator dibuat untuk training dibangun dengan memanfaatkan objek ImageDataGenerator yang sebelumnya telah dikonfigurasi dengan normalisasi piksel (rescale=1./255) dan augmentasi sederhana. Pemanggilan generator ini dilakukan menggunakan fungsi flow_from_dataframe() dengan sumber data berupa DataFrame yang telah melalui proses balancing.

```
train_gen = train_datagen.flow_from_dataframe(
    train_df_balanced,
    x_col='filepath',
    y_col='label',
    target_size=(IMG_SIZE, IMG_SIZE),
    class_mode='categorical',
    BATCH_SIZE=BATCH_SIZE,
    shuffle=True,
    seed=SEED,
    color_mode='rgb'

)
```

Gambar 3.20 Code Generator Train

Generator training pada gambar tersebut disusun untuk menyiapkan data citra agar siap digunakan oleh model selama proses pelatihan. Beberapa tahapan yang dilakukan dalam generator ini adalah sebagai berikut.

Pertama, dilakukan normalisasi citra, yaitu mengubah nilai piksel gambar ke rentang 0 hingga 1. Normalisasi ini bertujuan agar proses pembelajaran model menjadi lebih stabil dan konvergen, karena perbedaan skala nilai piksel dapat memengaruhi kinerja optimasi model. Kedua, data dibaca dalam bentuk batch. Setiap proses pelatihan tidak menggunakan seluruh data sekaligus, melainkan sebagian data sesuai dengan ukuran batch_size. Pendekatan ini membantu menghemat memori dan membuat proses training lebih efisien. Ketiga, data training diatur dalam kondisi acak (shuffle). Pengacakan ini dilakukan agar urutan data yang masuk ke model selalu berbeda pada setiap epoch, sehingga model tidak bergantung pada pola urutan tertentu dan mampu melakukan generalisasi dengan lebih baik. Selanjutnya, label kelas dikodekan dalam bentuk kategorikal (one-hot encoding). Format ini diperlukan karena model melakukan klasifikasi multi-kelas, sehingga setiap kelas direpresentasikan dalam vektor biner yang sesuai dengan keluaran fungsi softmax.

Terakhir, seluruh citra disesuaikan ukurannya dengan ukuran input model melalui parameter target_size. Penyeragaman ukuran ini diperlukan agar seluruh data dapat diproses secara konsisten oleh arsitektur jaringan saraf yang digunakan.

```
val_gen = val_datagen.flow_from_dataframe(
    val_df,
    x_col='filepath',
    y_col='label',
    target_size=(IMG_SIZE, IMG_SIZE),
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=False,
    seed=SEED,
    color_mode='rgb'
)
```

Gambar 3.21 Code Generator Validation

Augmentasi data tidak digunakan. Hal ini dilakukan untuk menjaga keaslian citra validation seperti pada gambar 3.21 sehingga hasil evaluasi benar-benar merefleksikan kemampuan model dalam mengenali data yang tidak pernah dilatih sebelumnya.

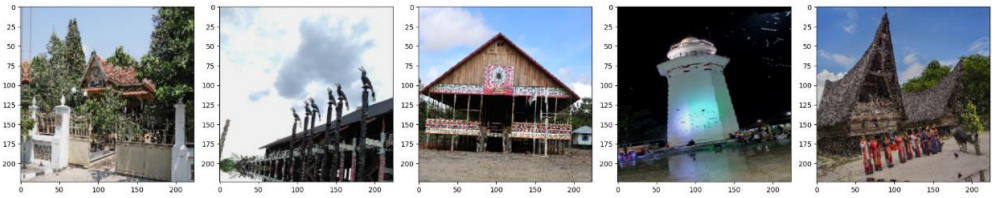
Kedua, data tidak diacak (shuffle=False). Pengaturan ini memastikan bahwa urutan data, label, dan hasil prediksi tetap konsisten, sehingga memudahkan proses analisis dan pelacakan hasil evaluasi model. Ketiga, normalisasi citra tetap diterapkan dengan skema yang sama seperti pada data training. Kesamaan proses normalisasi ini penting agar distribusi input antara data training dan validation tetap konsisten. Keempat, label kelas dikodekan dalam format kategorikal (one-hot encoding). Dengan format ini, hasil prediksi model dapat langsung dibandingkan dengan label sebenarnya yang dihasilkan oleh fungsi softmax pada lapisan output.

```
test_gen = test_datagen.flow_from_dataframe(
    dataframe=test_df,
    x_col="filepath",
    y_col=None,
    target_size=(IMG_SIZE, IMG_SIZE),
    class_mode=None,
    shuffle=False
)
```

Gambar 3.22 Code Generator Test Submission

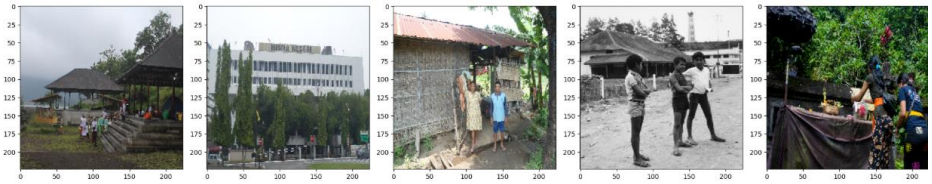
Selanjutnya generator pada gambar diatas untuk test submision dibuat untuk Pertama, data test tidak memiliki label (y_col=None) karena dataset ini digunakan untuk keperluan prediksi akhir dan penyusunan file submission. Oleh karena itu, parameter class_mode diset ke None.Kedua, tidak diterapkan augmentasi data, sehingga citra test diproses dalam kondisi aslinya. Pendekatan ini bertujuan agar hasil prediksi mencerminkan performa model pada data nyata. Ketiga, normalisasi

citra tetap dilakukan melalui rescale=1./255, sehingga distribusi input konsisten dengan data training dan validation. Keempat, data tidak diacak (shuffle=False). Hal ini penting untuk menjaga kesesuaian urutan antara file citra dan hasil prediksi model, terutama pada proses pembuatan file submission. Dan seluruh citra disesuaikan ukurannya dengan ukuran input model (target_size), sehingga dapat langsung diproses oleh arsitektur jaringan yang digunakan. Dan Selanjutnya pada gambar dibawah berikut contoh pada 5 gambar data pada train generator yang diambil;



Gambar 3.23 Sample Data Train Generator

Dan berikut ini sample yang diambil pada Validation generator seperti pada gambar dibawah;



Gambar 3.24 Sample Data Validation Generatort.

3.3.1.3 Tahap 3 Eksplorasi Model dan Evaluasi Lomba

Selanjutnya dibangun untuk Eksplorasi Model dan model yang digunakan pada lomba yaitu dengan menggunakan model MobileNetV2 seperti dibawah gambar berikut:

```
base_model = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(IMG_SIZE, IMG_SIZE, 3)
)
base_model.trainable = False  # Freeze semua layer di base model
# ====== CLASSIFIER HEAD ======
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(
    128, activation='relu',
    kernel_regularizer=regularizers.l2(0.001)
)(x)
x = Dropout(DROPOUT_RATE)(x)
predictions = Dense(
    num_of_classes,
    activation='softmax',
    kernel_regularizer=regularizers.l2(0.001)
)(x)
# ====== BUILD FINAL MODEL ======
model = Model(inputs=base_model.input, outputs=predictions)
# ====== KOMPILE MODEL ======
model.compile(
    optimizer=Adam(learning_rate=LEARNING_RATE),
    loss=tf.keras.losses.CategoricalCrossentropy(label_smoothing=0.1),
    metrics=['accuracy']
)
model.summary()
```

Gambar 3.25 Code Model Pretrained MobileNetV2

Pada tahap ini, dilakukan eksplorasi arsitektur *deep learning* dengan memanfaatkan model MobileNetV2 sebagai *pretrained model*. Pemilihan MobileNetV2 didasarkan pada kemampuannya dalam menyeimbangkan antara akurasi dan efisiensi komputasi, sehingga sesuai untuk kebutuhan klasifikasi citra dengan sumber daya terbatas. Model MobileNetV2 yang digunakan telah dilatih sebelumnya pada dataset ImageNet, sehingga memiliki kemampuan awal dalam mengekstraksi fitur visual tingkat rendah hingga menengah.

Pada tahap ini, MobileNetV2 digunakan sebagai *feature extractor* dengan parameter include_top=False, sehingga lapisan klasifikasi bawaan tidak disertakan. Seluruh bobot pretrained dari ImageNet dimanfaatkan untuk mengekstraksi fitur citra. Selanjutnya, seluruh lapisan pada *base model* dibekukan (*freeze*) untuk mencegah perubahan bobot selama pelatihan awal.

Output dari base model kemudian dilewatkan ke lapisan Global Average Pooling, yang berfungsi merangkum informasi spasial menjadi vektor fitur yang lebih ringkas. Pendekatan ini lebih stabil dibandingkan flatten layer dan membantu mengurangi jumlah parameter.

Lapisan Dense dengan 128 neuron dan fungsi aktivasi ReLU ditambahkan

untuk mempelajari pola non-linear dari fitur hasil ekstraksi. Regularisasi L2 diterapkan untuk mengurangi risiko overfitting, sementara Dropout digunakan untuk meningkatkan generalisasi model, lapisan output menggunakan fungsi aktivasi softmax, sesuai dengan permasalahan klasifikasi multi-kelas, sehingga menghasilkan probabilitas untuk setiap kelas rumah adat.

Model dikompilasi menggunakan optimizer Adam karena kemampuannya dalam mempercepat konvergensi. Fungsi loss yang digunakan adalah Categorical Crossentropy dengan label smoothing untuk meningkatkan stabilitas pembelajaran dan mengurangi kepercayaan berlebih model terhadap satu kelas tertentu.

```python
from sklearn.utils.class_weight import compute_class_weight
import numpy as np

# Hitung bobot kelas berdasarkan data train_df_balanced
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_df_balanced['label']),
    y=train_df_balanced['label']
)

# Konversi ke dict dengan indeks sesuai urutan label di generator
label2index = {label: idx for idx, label in enumerate(train_gen.class_indices)}
class_weights = {label2index[cls]: weight for cls, weight in zip(np.unique(train_df_balanced['label']), class_weights)}

print("Class weights:")
for k, v in class_weights.items():
    print(f"{k}: {v:.3f}")
```

Gambar 3.26 Code Ketidakseimbangan Kelas dengan Class Weight

Selanjutnya setelah model dikompilasi dilakukan pendekatan maka akan diterapkan Class Weight seperti gambar 3.26 Meskipun data telah melalui proses *soft balancing*, bobot kelas tetap dihitung untuk memberikan perhatian lebih pada kelas minoritas selama pelatihan. Bobot ini kemudian dipetakan ke indeks kelas sesuai urutan label pada generator training. Pendekatan ini membantu model mempelajari seluruh kelas secara lebih adil dengan bobot kelas seperti gambar dibawah berikut;

```
Class weights:
0: 0.900
1: 1.080
2: 1.080
3: 1.080
4: 0.900
```

Gambar 3.27 Bobot Class yang sudah di Balanced

Dan Jumlah langkah pelatihan ditentukan berdasarkan jumlah sampel dan ukuran batch: Pelatihan awal dilakukan dengan callback sebagai berikut EarlyStopping untuk menghentikan training jika performa validation tidak meningkat, ReduceLROnPlateau untuk menurunkan learning rate ketika validation loss stagnan, ModelCheckpoint untuk menyimpan model terbaik berdasarkan validation loss. Sebagai contoh gambar dibawah berikut :

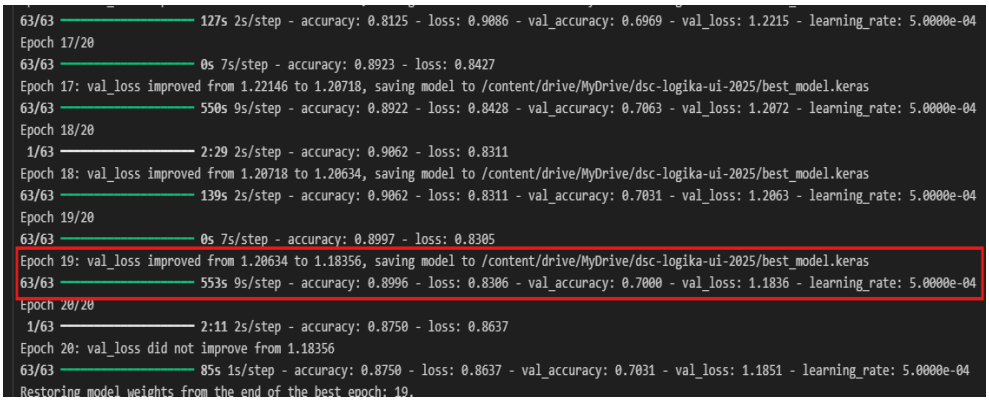
```
steps_per_epoch = (train_gen.samples // train_gen.batch_size)
val_steps = (val_gen.samples // val_gen.batch_size)

print("Steps per epoch:", steps_per_epoch)
print("Validation steps:", val_steps)
```

Gambar 3.28 Jumlah langkah pelatihan sampel

```
Steps per epoch: 63
Validation steps: 10
```

Gambar 3.29 Jumlah pelatihan train dan validasi

Jumlah pada gambar diatas ini selanjutnya akan duji bertujuan melatih classifier head tanpa mengubah bobot pretrained MobileNetV2 dan hasil pengujian berikut didapati sebagai gambar berikut;



Gambar 3.30 Hasil Traning Model

Berdasarkan hasil pelatihan, epoch ke-19 tercatat sebagai epoch terbaik, karena menghasilkan nilai validation loss terendah dibandingkan epoch lainnya. Hal ini ditunjukkan oleh mekanisme ModelCheckpoint yang menyimpan bobot model pada epoch tersebut. Performa Model pada Epoch Terbaik (Epoch 19), penurunan nilai validation loss yang konsisten hingga epoch ke-19 mengindikasikan bahwa model mampu mempelajari pola penting dari data tanpa mengalami overfitting yang signifikan. Meskipun terdapat selisih antara akurasi training dan validation, perbedaan tersebut masih berada dalam batas wajar untuk dataset dengan distribusi kelas yang tidak seimbang. Setelah model mencapai performa yang stabil, dilakukan fine-tuning dengan membuka sebagian lapisan atas MobileNetV2 seperti gambar dibawah ini;

```
print("Unfreezing top layers for fine-tuning...")
base_model.trainable = True
# freeze all but last N layers (tweak N as desired)
N = 20
for layer in base_model.layers[:-N]:
    layer.trainable = False
for layer in base_model.layers[-N:]:
    layer.trainable = True
```

Gambar 3.31 Code Fine Tuning

Hanya 20 lapisan terakhir yang terlihat pada gambar 3.31 yang dilatih ulang, sementara lapisan awal tetap dibekukan. Pendekatan ini memungkinkan model menyesuaikan fitur tingkat tinggi dengan karakteristik dataset tanpa merusak fitur dasar yang telah dipelajari. Dan learning rate diturunkan secara signifikan untuk

menjaga kestabilan pembelajaran:

```
model.compile(
    optimizer=Adam(learning_rate=1e-5),
    loss=tf.keras.losses.CategoricalCrossentropy(label_smoothing=0.1),
    metrics=['accuracy']
)
```

Gambar 3.32 Learning rate pada fine Tuning

Model kemudian dikompilasi ulang dengan learning rate kecil (1e-5) agar pembaruan bobot berlangsung stabil seperti terlihat gambar 3.32 dan menghasilkan fine tuning sebagai berikut ;

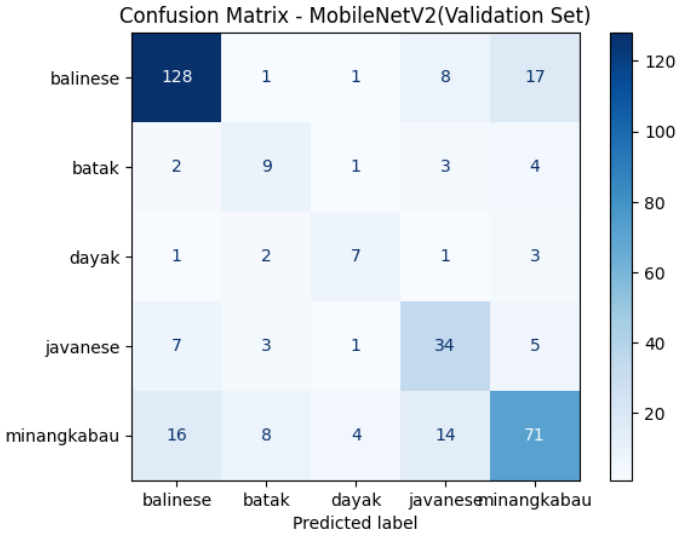
```
Unfreezing top layers for fine-tuning...
Epoch 1/15
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.8482 - loss: 0.9096
Epoch 1: val_loss did not improve from 1.18356
63/63 ━━━━━━━━━━━━━━━━━━━━ 597s 9s/step - accuracy: 0.8484 - loss: 0.9094 - val_accuracy: 0.7094 - val_loss: 1.1880 - learning_rate: 1.0000e-05
Epoch 2/15
 1/63 ━━━━━━━━━━━━━━━━━━━━ 7:07 7s/step - accuracy: 0.8438 - loss: 0.9519
Epoch 2: val_loss did not improve from 1.18356
63/63 ━━━━━━━━━━━━━━━━━━━━ 100s 1s/step - accuracy: 0.8438 - loss: 0.9519 - val_accuracy: 0.7063 - val_loss: 1.1882 - learning_rate: 1.0000e-05
Epoch 3/15
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 8s/step - accuracy: 0.8667 - loss: 0.8646
Epoch 3: val_loss did not improve from 1.18356
63/63 ━━━━━━━━━━━━━━━━━━━━ 601s 9s/step - accuracy: 0.8666 - loss: 0.8646 - val_accuracy: 0.7063 - val_loss: 1.1917 - learning_rate: 1.0000e-05
Epoch 4/15
 1/63 ━━━━━━━━━━━━━━━━━━━━ 2:42 3s/step - accuracy: 0.9062 - loss: 0.8110
Epoch 4: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-06.

Epoch 4: val_loss did not improve from 1.18356
63/63 ━━━━━━━━━━━━━━━━━━━━ 89s 1s/step - accuracy: 0.9062 - loss: 0.8110 - val_accuracy: 0.7063 - val_loss: 1.1916 - learning_rate: 1.0000e-05
Epoch 5/15
```

Gambar 3.33 Hasil fine tuning

Selama proses fine-tuning, model menunjukkan peningkatan performa yang signifikan pada data training, ditandai dengan kenaikan akurasi yang konsisten hingga mencapai sekitar 90% serta penurunan nilai loss, yang mengindikasikan bahwa bobot pada layer teratas MobileNetV2 mampu beradaptasi dengan baik terhadap pola data. Pada epoch awal, akurasi training berada pada kisaran 84,8% (epoch 1), kemudian meningkat menjadi sekitar 90,6% pada epoch 4, dan bahkan mencapai lebih dari 96% pada beberapa batch di epoch 6, yang menunjukkan kemampuan representasi fitur tingkat tinggi yang semakin optimal. Namun demikian, performa pada data validation relatif stabil dengan akurasi berada pada rentang 70–72% dan nilai validation loss yang tidak mengalami perbaikan signifikan sejak epoch awal, sehingga dapat disimpulkan bahwa peningkatan performa pada data training tidak sepenuhnya tertransfer ke data validation. Kondisi ini mengindikasikan adanya kecenderungan overfitting ringan, meskipun telah diterapkan teknik regularisasi dan class weighting. Dalam proses training,

callback berperan penting dalam menjaga stabilitas model, khususnya ReduceLROnPlateau yang menurunkan learning rate secara bertahap dari 1e-5 menjadi 5e-6 hingga 2.5e-6 ketika validation loss tidak membaik, sehingga proses optimisasi menjadi lebih halus dan terkontrol. Selain itu, mekanisme Early Stopping menghentikan training pada epoch ke-7 karena tidak adanya peningkatan validation loss, serta mengembalikan bobot model terbaik dari epoch ke-1 yang memiliki nilai validation loss terendah, sehingga model akhir yang digunakan merupakan model dengan kemampuan generalisasi terbaik, bukan sekadar model dengan akurasi training tertinggi.

Tahap evaluasi model dilakukan menggunakan confusion matrix, classification report, dan F1-score (macro) pada data validation untuk memperoleh gambaran performa model secara menyeluruh, khususnya pada kondisi dataset yang tidak seimbang.

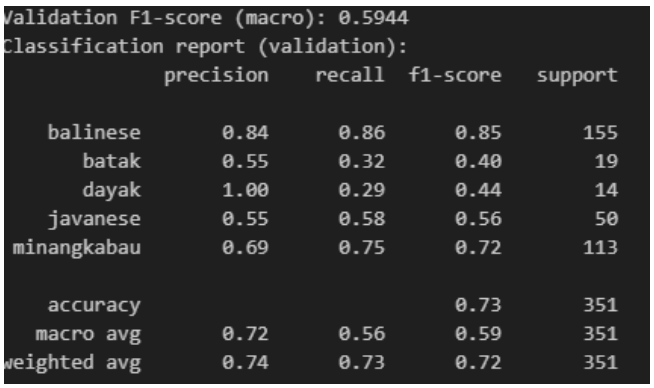


Gambar 3.34 Hasil Matrix evaluated MobileNetV2

Berdasarkan confusion matrix gambar diatas, terlihat bahwa model mampu mengklasifikasikan kelas **balinese** dengan sangat baik, ditunjukkan oleh jumlah prediksi benar yang tinggi (128 dari 155 sampel). Hal ini sejalan dengan nilai precision dan recall yang sama-sama mencapai 0,83. Kelas **minangkabau** juga menunjukkan performa yang relatif stabil dengan f1-score sebesar 0,67, meskipun masih terdapat beberapa kesalahan prediksi ke kelas lain, terutama ke kelas

javanese dan balinese.

Sebaliknya, performa pada kelas dengan jumlah data lebih sedikit, seperti batak dan dayak, masih tergolong lebih rendah. Kelas batak memiliki f1-score sebesar 0,43 dan kelas dayak sebesar 0,50. Hal ini menunjukkan bahwa meskipun telah diterapkan class weighting dan stratified split, model masih mengalami kesulitan dalam membedakan pola visual pada kelas minoritas, yang merupakan tantangan umum pada dataset imbalanced.



```
Validation F1-score (macro): 0.5944
Classification report (validation):
              precision    recall  f1-score   support

    balinese       0.84      0.86      0.85       155
       batak       0.55      0.32      0.40        19
       dayak       1.00      0.29      0.44        14
    javanese       0.55      0.58      0.56        50
 minangkabau       0.69      0.75      0.72       113

    accuracy                           0.73       351
   macro avg       0.72      0.56      0.59       351
weighted avg       0.74      0.73      0.72       351
```

Gambar 3.35 Hasil Uji Model MobileNetV2

Secara keseluruhan, model mencapai akurasi validation pada gambar diatas sebesar 71%, dengan F1-score macro sebesar 0,6084. Nilai F1 macro ini menjadi indikator penting karena menghitung rata-rata performa setiap kelas secara setara, sehingga memberikan gambaran yang lebih adil terhadap performa model pada kelas mayoritas maupun minoritas. Hasil ini menunjukkan bahwa model telah memiliki kemampuan klasifikasi yang cukup baik, namun masih memiliki ruang untuk peningkatan, khususnya pada kelas dengan jumlah sampel terbatas.

3.3.1.4 Tahap 4 Finalisasi Model dan Submission Lomba

Setelah model MobileNetV2 terbaik diperoleh berdasarkan nilai validation loss terendah, tahap selanjutnya adalah melakukan inferensi pada data test yang tidak memiliki label. Proses ini bertujuan untuk menghasilkan prediksi kelas akhir yang akan digunakan sebagai hasil submission pada kompetisi.

```python
print("Predicting test set...")
test_steps = len(test_df)  # since batch_size=1
pred_probs = model.predict(test_generator, steps=test_steps, verbose=1)
pred_classes = np.argmax(pred_probs, axis=1)
# Map index -> class label string
labels_map = {v: k for k, v in train_gen.class_indices.items()}
pred_labels = [labels_map[i] for i in pred_classes]
# Build submission DataFrame
sample_submission = pd.read_csv(SAMPLE_SUB_PATH)
submission = pd.DataFrame({
    "id": test_df['id'],
    "style": pred_labels
})
# Reindex to follow sample_submission order (if column 'id' exists)
if 'id' in sample_submission.columns:
    submission = submission.set_index('id').reindex(sample_submission['id']).reset_index()
# Save final submission
SAVE_PATH = "/content/drive/MyDrive/dsc-logika-ui-2025/DSC0024_Anamorphic_Submisi.csv"
submission.to_csv(SAVE_PATH, index=False)

print(f"✅ Submission saved to: {SAVE_PATH}")
print("First rows of submission:")
print(submission.head())
```

Gambar 3.36 Prediksi Data Uji dan Penyusunan File Submission

Prediksi dilakukan menggunakan test_generator seperti gambar diatas dengan pengaturan shuffle=False dan batch_size=1, sehingga urutan prediksi tetap konsisten dengan urutan file citra pada data test. Model menghasilkan probabilitas untuk setiap kelas, kemudian kelas dengan nilai probabilitas tertinggi dipilih sebagai label prediksi menggunakan fungsi argmax. Hasil prediksi indeks kelas selanjutnya dipetakan kembali ke nama label aslinya (balinese, batak, dayak, javanese, dan minangkabau) berdasarkan class indices yang digunakan pada proses training.

```
Predicting test set...
444/444 ━━━━━━━━━━━━━━━━━━━━ 146s 318ms/step
✅ Submission saved to: /content/drive/MyDrive/dsc-logika-ui-2025/DSC0024_Anamorphic_Submisi.csv
First rows of submission:
        id        style
0  Test_001     balinese
1  Test_002  minangkabau
2  Test_003     javanese
3  Test_004     balinese
4  Test_005  minangkabau
```

Gambar 3.37 Format Submission

Agar format submission sesuai dengan ketentuan kompetisi, hasil prediksi disusun ke dalam sebuah DataFrame dengan dua kolom utama, yaitu id dan style. Urutan id kemudian disesuaikan dengan sample submission resmi untuk memastikan tidak terjadi pergeseran data. File submission akhir disimpan dalam format CSV dan berhasil dihasilkan tanpa error, yang menandakan bahwa pipeline inferensi telah berjalan dengan benar dan siap untuk proses pengunggahan.

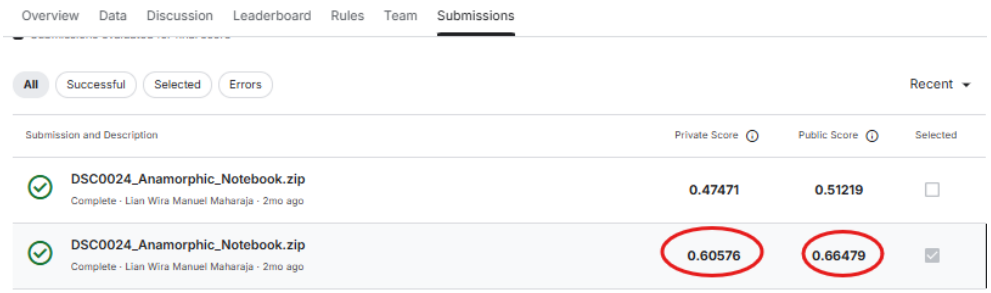
Berdasarkan evaluasi pada data validation, model mencapai akurasi sebesar ±71–73%, dengan F1-score macro berkisar antara 0,59–0,61. Nilai F1 macro digunakan sebagai metrik utama karena memberikan bobot yang setara pada setiap kelas, sehingga lebih representatif untuk dataset dengan distribusi kelas yang tidak seimbang. Hasil classification report menunjukkan bahwa model memiliki performa sangat baik pada kelas balinese dan minangkabau, dengan nilai precision dan recall yang relatif tinggi. Sebaliknya, performa pada kelas minoritas seperti batak dan dayak masih lebih rendah, yang tercermin dari nilai recall dan f1-score yang terbatas. Temuan ini konsisten dengan analisis confusion matrix, di mana kesalahan klasifikasi lebih sering terjadi pada kelas dengan jumlah data yang lebih sedikit. Secara keseluruhan, hasil evaluasi menunjukkan bahwa model MobileNetV2 mampu menangkap pola visual utama pada dataset dengan baik, namun masih menghadapi tantangan dalam membedakan kelas-kelas minoritas. Meskipun demikian, penerapan transfer learning, class weighting, dan fine-tuning telah memberikan peningkatan performa yang signifikan dibandingkan pendekatan baseline

3.3.1.5 Tahap 5 Hasil Lomba/Kompetisi

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

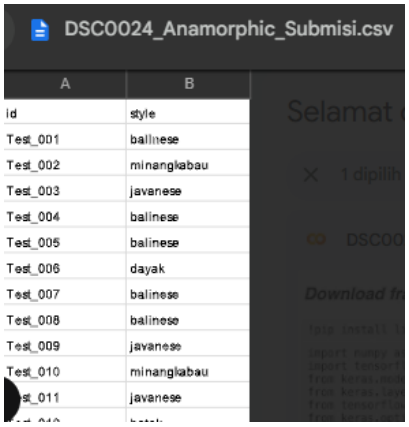
Gambar 3.38 menunjukkan hasil leaderboard setelah model dikumpulkan pada halaman kompetisi LOGIKA UI 2025 di Kaggle.com. Berdasarkan hasil tersebut, tim peneliti memperoleh peringkat ke-60 dari total 94 kelompok peserta.

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 3.38 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 3.39.



Gambar 3.39 Hasil CSV

Sebagai bentuk apresiasi atas partisipasi dalam kompetisi, penyelenggara Data Science Competition (DSC) LOGIKA UI 2025 memberikan sertifikat keikutsertaan kepada peserta. Sertifikat tersebut menjadi bukti bahwa tim telah mengikuti dan menyelesaikan seluruh rangkaian kompetisi secara resmi, sebagaimana ditunjukkan pada Gambar 3.40.

Gambar 3.40 Sertifikasi Kompetisi.

Sertifikat pada gambar 3.40 mencakup detail seperti nama peserta, judul proyek, program studi, dan institusi asal, yang dikeluarkan oleh pihak instansi lomba sebagai bagian dari program Road to Champion. Sertifikat ini tidak hanya menjadi pengakuan atas upaya yang telah dilakukan, tetapi juga sebagai motivasi untuk terus mengembangkan kemampuan di bidang data science dan machine learning. Keikutsertaan dalam kompetisi ini memberikan pengalaman berharga dalam menghadapi tantangan nyata.

3.3.1.6 Tahap 6 Lanjutan Esplorasi Pengembangan dan Optimasi Model

Selanjutnya proses lanjutan eksplorasi pada Gambar 3.41 menunjukkan hasil pembagian dataset citra ke dalam data latih (training), validasi (validation), dan uji (testing) menggunakan metode stratified split. Dataset terdiri dari 1.752 citra yang terbagi ke dalam lima kelas, yaitu Balinese, Minangkabau, Javanese, Batak, dan Dayak.

```
train_df, temp_df = train_test_split(
    df,
    test_size=0.2,
    stratify=df['label'],
)

print("Train size:", len(train_df))
print("Temp size :", len(temp_df))

val_df, sample = train_test_split(
    temp_df,
    test_size=1/3,
    stratify=temp_df['label'],
)

print("Val size :", len(val_df))
print("sample size:", len(sample))
```

Gambar 3.41 Split yang diperbarui

Proses pembagian data dilakukan dalam dua tahap. Pada tahap pertama, dataset dibagi menjadi 80% data latih dan 20% data sementara (temporary set). Selanjutnya, data sementara tersebut dibagi kembali menjadi data validasi (≈13%) dan data uji (≈7%).

Pendekatan stratified split diterapkan untuk memastikan proporsi masing-masing kelas tetap terjaga pada setiap subset data. Hasil pembagian menunjukkan bahwa distribusi kelas pada data latih, validasi, dan uji tetap konsisten dengan distribusi dataset awal, sehingga dapat meminimalkan bias selama proses pelatihan dan evaluasi model seperti gambar dibawah ini;

```
Total images: 1752
label
balinese       776
minangkabau    563
javanese       249
batak           95
dayak           69
Name: count, dtype: int64
Train size: 1401
Temp size : 351
Val size : 234
sample size: 117
Found 1401 validated image filenames belonging to 5 classes.
Found 234 validated image filenames belonging to 5 classes.
Found 444 validated image filenames.
```

Gambar 3.42 Hasil Split yang diperbarui tanpa Balancing Training

Penelitian ini menggunakan arsitektur MobileNetV2, ResNet50V2, DenseNet121, VGG16 sebagai base model dengan bobot awal ImageNet. Model digunakan dalam skema transfer learning dengan menghilangkan lapisan klasifikasi bawaan (include_top = False) dan membekukan seluruh lapisan pada base model untuk mempertahankan fitur visual tingkat rendah dan menengah.

Sebagai classifier head, ditambahkan lapisan Global Average Pooling untuk mereduksi dimensi fitur, diikuti oleh lapisan Dense dengan 128 neuron dan fungsi

aktivasi ReLU, serta Dropout untuk mengurangi risiko overfitting. Lapisan keluaran menggunakan Dense dengan fungsi aktivasi softmax untuk menghasilkan probabilitas prediksi pada lima kelas.

Model dikompilasi menggunakan optimizer Adam, fungsi kehilangan Categorical Crossentropy, dan metrik evaluasi akurasi. Pendekatan ini bertujuan untuk memperoleh model yang efisien secara komputasi dengan performa klasifikasi yang optimal.

Tabel 3.2 Tabel Perbandingan Kinerja Model Klasifikasi Tanpa Data Balancing

Model	Data Balancing	Accuracy Train (%)	Accuracy Val (%)	Loss Val	F1-score (Macro)	Keterangan
VGG16	Tidak	68.60	56.01	1.1350	0.4590	Cenderung overfitting, performa rendah pada kelas minoritas
MobileNetV2	Tidak	67.43	66.07	0.9206	0.5301	Stabil, ringan, namun lemah pada kelas minoritas
ResNet50V2	Tidak	85.93	71.88	0.8406	0.6349	Generalisasi baik, performa meningkat signifikan
DenseNet121	Tidak	85.45	87.19	0.3356	0.8808	Performa terbaik, sangat stabil di semua kelas

Berdasarkan hasil pengujian pada tabel 3.2 diatas tanpa penerapan teknik data balancing, terlihat bahwa setiap arsitektur deep learning menunjukkan karakteristik performa yang berbeda. Model VGG16 menghasilkan akurasi validasi terendah sebesar 56,01% dengan nilai macro F1-score 0,4590, yang mengindikasikan keterbatasan model dalam mengenali kelas minoritas pada dataset yang tidak seimbang. Hal ini juga tercermin dari nilai validation loss yang relatif tinggi.

Model MobileNetV2 menunjukkan peningkatan performa dibandingkan VGG16 dengan akurasi validasi 66,07% dan macro F1-score 0,5301. Meskipun lebih stabil dan efisien secara komputasi, model ini masih mengalami kesulitan dalam mengklasifikasikan kelas dengan jumlah data terbatas, seperti Batak dan Dayak.

Sementara itu, ResNet50V2 memberikan performa yang lebih baik dengan akurasi validasi 71,88% dan macro F1-score 0,6349. Arsitektur residual yang digunakan memungkinkan model untuk mengekstraksi fitur yang lebih kompleks, sehingga meningkatkan kemampuan generalisasi dibandingkan VGG16 dan

MobileNetV2.

Model DenseNet121 menunjukkan performa terbaik di antara seluruh model yang diuji. Model ini mencapai akurasi validasi tertinggi sebesar 87,19%, nilai validation loss terendah 0,3356, serta macro F1-score 0,8808. Hasil ini menunjukkan bahwa mekanisme dense connectivity pada DenseNet mampu mempertahankan informasi fitur secara lebih efektif dan memberikan performa yang sangat baik, termasuk pada kelas minoritas.

Ketidakseimbangan data terbukti memengaruhi performa model, khususnya pada kelas dengan jumlah sampel yang terbatas. Model dengan arsitektur yang lebih kompleks dan konektivitas fitur yang kuat, seperti DenseNet121 dan ResNet50V2 sebagai gambar berikut

```
Validation F1-score (macro): 0.8808
Classification report (validation):
              precision    recall  f1-score   support

    balinese       0.96      0.86      0.91       154
       batak       0.72      0.95      0.82        19
       dayak       1.00      1.00      1.00        13
    javanese       0.74      0.88      0.80        49
 minangkabau       0.87      0.88      0.87       112

    accuracy                           0.88       347
   macro avg       0.86      0.91      0.88       347
weighted avg       0.89      0.88      0.88       347
```

Gambar 3.43 Hasil Pengujian DenseNet121

```
Validation F1-score (macro): 0.6349
Classification report ResNet50V2 (validation):
              precision    recall  f1-score   support

    balinese       0.88      0.75      0.81       104
       batak       0.33      0.62      0.43        13
       dayak       0.58      0.78      0.67         9
    javanese       0.51      0.64      0.57        33
 minangkabau       0.74      0.67      0.70        75

    accuracy                           0.70       234
   macro avg       0.61      0.69      0.63       234
weighted avg       0.74      0.70      0.71       234
```

Gambar 3.44 Hasil Pengujian ResNet50V2

Mampu mengatasi permasalahan ini dengan lebih baik dibandingkan model yang lebih sederhana. Rendahnya nilai *recall* dan *F1-score* pada kelas minoritas pada VGG16 dan MobileNetV2 pada gambar dibawah ini menunjukkan bahwa model tersebut cenderung bias terhadap kelas mayoritas. Sebaliknya, DenseNet121 mampu mempertahankan performa yang seimbang antar kelas, yang ditunjukkan oleh nilai *macro F1-score* yang tinggi.

```
Validation F1-score (macro): 0.5301
Classification report MobileNetV2 (validation):
              precision    recall  f1-score   support

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

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

Gambar 3.45 Hasil Pengujian MobileNetV2 Tanpa Balancing

```
Validation F1-score (macro): 0.4590
Classification report (validation):
              precision    recall  f1-score   support

    balinese       0.70      0.69      0.69       194
       batak       0.28      0.35      0.31        23
       dayak       0.20      0.50      0.28        18
    javanese       0.43      0.48      0.46        62
 minangkabau       0.65      0.48      0.56       141

    accuracy                           0.57       438
   macro avg       0.45      0.50      0.46       438
weighted avg       0.60      0.57      0.58       438
```

Gambar 3.46 Hasil Pengujian VGG16

Penerapan teknik *data balancing* pada MobileNetV2 menunjukkan peningkatan performa khususnya pada kelas minoritas. Hal ini tercermin dari meningkatnya nilai *macro F1-score* dan *recall* pada kelas dengan jumlah data terbatas. Dengan demikian, data balancing terbukti efektif dalam mengurangi bias prediksi pada model berbasis *transfer learning* yang ringan seperti MobileNetV2.

Berdasarkan hasil eksperimen, dapat disimpulkan bahwa DenseNet121 merupakan model terbaik dalam penelitian ini pada skenario tanpa data balancing. Model ini tidak hanya mencapai akurasi tertinggi, tetapi juga menunjukkan kestabilan prediksi yang sangat baik pada seluruh kelas. Oleh karena itu, DenseNet121 dipilih sebagai model utama untuk tahap pengujian lanjutan dan analisis lebih

3.3.2 Kendala yang Ditemukan

Selama proses pengembangan dan pengujian model klasifikasi citra rumah adat di Indonesia, terdapat beberapa kendala yang dihadapi, baik pada tahap pengolahan data maupun pada tahap pemodelan. Kendala-kendala tersebut

diuraikan sebagai berikut:

1. Ketidakseimbangan Dataset

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

2. Performa Model yang Tidak Stabil pada Awal Training

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

3. Waktu Eksperimen yang Relatif Panjang

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

3.3.3 Solusi atas Kendala yang Ditemukan

Selama proses pengembangan dan pengujian model klasifikasi citra rumah adat di Indonesia, terdapat beberapa kendala yang dihadapi, baik pada tahap pengolahan data maupun pada tahap pemodelan. Kendala-kendala tersebut diuraikan sebagai berikut:

1. Ketidakseimbangan Dataset

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

2. Performa Model yang Tidak Stabil pada Awal Training

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

3.Waktu Eksperimen yang Relatif Panjang

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