BAB III

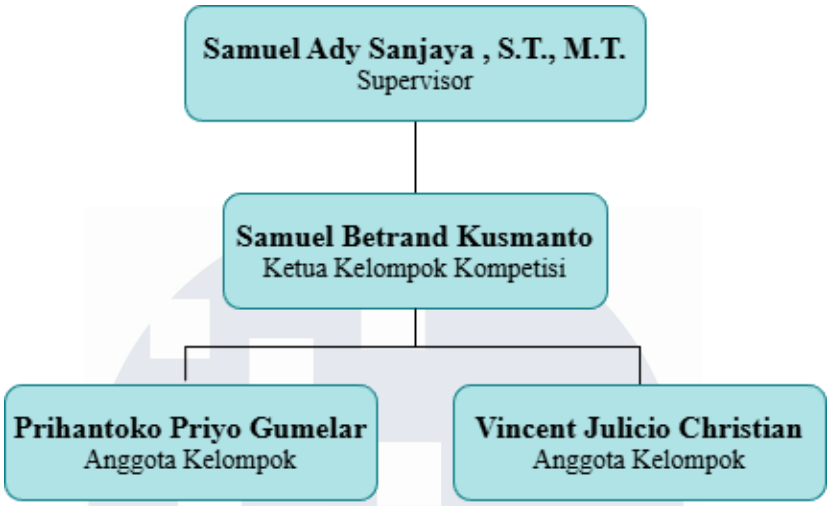
PELAKSANAAN PRO-STEP: ROAD TO CHAMPION

3.1 Kedudukan dan Koordinasi

Pelaksanaan PRO-STEP: Road To Champion yang dilaksanakan oleh 3 mahasiswa yang berkelompok dengan dibimbing oleh 2 dosen pembimbing untuk masalah teknis lomba, seperti pemberian solusi dan arahan dalam pengerjaan kompetisi dari awal dan akhir sehingga dapat berjalan dengan lancar dan mendapatkan hasil yang baik dan memuaskan dibimbing oleh Bapak Samuel Ady Sanjaya , S.T., M.T. Sebagai anggota kelompok selama pelaksanaan PRO-STEP: Road To Champion tugas yang dijalankan membantu ketua kelompok dalam menyusun strategi pengerjaan kompetisi dan mengimplementasikan strategi yang sudah disepakati sehingga mendapatkan hasil yang baik dan memuaskan bagi kompetisi. Sebagai anggota kelompok juga wajib untuk selalu aktif untuk memberikan kontribusi baik dalam *update progress* dalam menjalani *code* dan hasil yang didapatkan dari menjalankan *code* kepada anggota kelompok.

1) **Kedudukan Antara Dosen Pembimbing Kompetisi dengan Kelompok / Individu Peserta Kompetisi Akademik / Beberapa pihak yang Berkepentingan**

   Dibawah ini merupakan bagan alur yang menjadi penjabaran kedudukan antara supervisor dan kelompok dalam pelaksanaan PRO-STEP: Road To Champion dan *jobdesc* dari setiap kedudukan dari supervisor sampai anggota kelompok.



Gambar 3. 1 Bagan Alur Koordinasi

Berdasarkan gambar 3.1 diatas yang merupakan bagan alur koordinasi dari supervisor, ketua kelompok dan anggota kelompok. Peran supervisor sebagai pembimbing yang mengarahkan solusi dan proses pengerjaan yang baik selama kompetisi berlangsung. Supervisor juga memberikan evaluasi seperti solusi, arahan dan ide untuk memperbaiki dari hasil yang sudah didapatkan sehingga ketikan menjalankan kode kembali mendapatkan hasil yang maksimal. Peran ketua kelompok yang utama adalah memastikan koordinasi antara supervisor dan anggota disampaikan dengan baik dan jelas. Ketua kelompok juga bertanggung jawab untuk memantau progress dari anggota kelompok sehingga dapat melaporkan hasil pengerjaan yang sudah kelompok kerjakan kepada supervisor, dan evaluasi dari supervisor disampaikan kembali ke anggota kelompok. Peran anggota kelompok adalah untuk membantu mengimplementasi strategi pengerjaan yang sudah didiskusikan bersama ketua kelompok dalam pengerjaan kompetisi, melaporkan dan menyelesaikan secara bersama untuk setiap masalah yang ditemukan selama pengerjaan kompetisi. Anggota kelompok juga wajib untuk aktif melaporkan hasil yang sudah didapat dari pengerjaan kepada ketua kelompok untuk mendapatkan evaluasi, solusi, arahan, dan saran dari

supervisor untuk hasil yang lebih maksimal.

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

Berikut merupakan tabel untuk detail pekerjaan yang dilakukan dalam *PRO-STEP: Road To Champion Program* dari minggu awal kompetisi dimulai, pengerjaan kompetisi hingga pengumpulan file.

Tabel 3. 1 Detail Pekerjaan yang dilakukan PRO-STEP: Road to Champion Program

No.	Minggu	Proyek	Keterangan
1	1	Mendapatkan brief lomba	Mendapatkan brief lomba dari penyelenggara lomba melalui zoom
2	1	Brainstorming pengerjaan lomba dengan kelompok	Brainstorming mengenai strategi yang akan di lakukan untuk pengerjaan kompetisi dengan kelompok
3	1	Berdiskusi untuk menentukan model yang akan dipakai	Memahami dan mempelajari setiap model yang memungkinkan untuk dipakai dan memutuskan model apa saja yang akan dipakai dalam pengerjaan kompetisi
4	2-3	Mengimplementasikan model yang sudah disepakati	Menjalankan kode dengan beberapa model *image classification* untuk mendapatkan hasil akurasi yang tinggi
5	2-3	Melaporkan hasil yang didapatkan kepada supervisor	Melaporkan hasil sementara yang didapatkan secara berkala, sehingga mendapatkan solusi dan arahan untuk memperbaiki hasil lebih baik
6	3	Mengumpulkan hasil dari pengerjaan kompetisi yang memiliki akurasi tertinggi ke kaggle	Mengumpulkan hasil dari pengerjaan yang terbagi menjadi 2 file yaitu file kode dan file prediksi yang berupa .ipynb dan .csv

Berdasarkan tabel 3.1 yang merupakan detail pengerjaan yang dilakukan selama pelaksanaan *PRO-STEP: Road To Champion* dari awal mendapatkan *briefing* dari pihak panitia LOGIKA UI 2025, berdiskusi mengenai strategi pengerjaan seperti pemilihan model, pembagian tugas, dan output yang akan dihasilkan, mengimplementasikan strategi pengerjaan dengan model yang sudah disepakati, melaporkan hasil secara berkala terhadap supervisor untuk mendapatkan evaluasi sehingga mendapatkan hasil yang lebih baik dan di akhir

deadline mengumpulkan hasil dengan tingkat akurasi yang tinggi yang sudah didapatkan. Berikut merupakan tabel tahapan kegiatan yang dilakukan dari mulai mempersiapkan dan melaksanakan kompetisi dari bulan Agustus sampai bulan Oktober pada PROSTEP: Road to Champion:

Tabel 3. 2 Detail Kegiatan selama pelaksanaan PROSTEP: Road to Champion

Kegiatan	Agustus			September					Oktober	
	3	4	5	1	2	3	4	5	1	2
Mereview Materi Perkuliahan Data Analysis										
Mereview Materi Perkuliahan Database										
Mereview Materi Perkuliahan Machine Learning										
Mereview Materi Perkuliahan DeepLearning										
Trial and Error Materi yang sudah direview										
Mempelajari materi Image Classification										
Menjalankan Model Inception V3										
Menjalankan Hybrid Model Convnext Tiny dengan EfficientNet-B0										
Menjalan Model ResNet 34 dengan finetuning										
Menjalankan Hybrid Model ResNet 34 dengan Swin Transformer										
Menjalankan Model EfficientNet-B1										
Menjalankan Model ResNet 50										

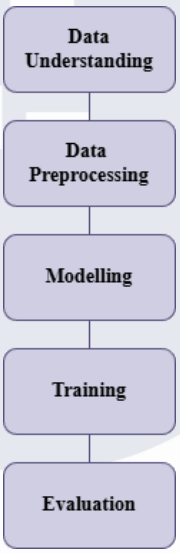
3.3 Uraian Pelaksanaan Kerja Dalam *PRO-STEP : Road To Champion Program*

Bagian ini berupa penjelasan secara umum mengenai pekerjaan yang dilakukan selama proses pelaksanaan dari *Data Understanding, Data Preprocessing, Modelling, Training, Evaluation* dalam melaksanakan *PRO-STEP: Road to Champion Program.*

3.3.1 Proses Pelaksanaan

Dalam proses pelaksanaan PRO-STEP : *Road To Champion* terdapat penjelasan dari tahapan yang dilaksanakan selama proses pengerjaan kompetisi LOGIKA UI 2025. Kompetisi ini menguji para mahasiswa untuk merancang sistem yang mengklasifikasikan gambar budaya di Indonesia yang ada di berbagai

daerah di Indonesia yang berfokus hanya pada 5 adat. Model utama yang digunakan adalah *EfficientNet-B1*, alur pengembangan model klasifikasi gambar umumnya sama, yaitu dimulai dari *data understanding, data preprocessing, modeling, training*, dan *evaluation*. Gambar dibawah adalah alur proses pelaksanaan pengerjaan pembangunan model klasifikasi gambar untuk kompetisi LOGIKA UI 2025.

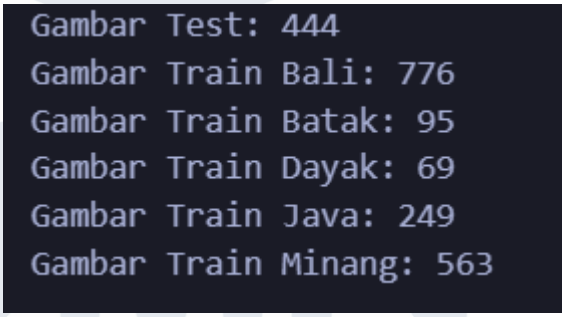


Gambar 3. 2 Proses Pelaksanaan Pengerjaan

Berdasarkan gambar 3.2 diatas merupakan proses pelaksanaan pengerjaan kompetisi yang di bagi menjadi 5 tahapan, tahapan tersebut adalah *data understanding, data preprocessing, modelling, training,* dan *evaluation.* uraian berfokus pada beberapa bagian pekerjaan yang dilakukan dalam PRO-STEP : *Road To Champion Program* jalur lomba/kompetisi. Pada Sub Bab ini, terdapat rincian dari langkah-langkah yang dilakukan selama pengerjaan lomba/kompetisi. Banyaknya detail/langkah-langkah yang akan dijabarkan harap dikonsultasikan dengan dosen pembimbing. Jika dirasa perlu mencantumkan bagan/skema alur pengerjaan silakan buat bagan/skema tersebut. Penjelasan pada bagian ini harus rinci dan menggambarkan apa yang dikerjakan. Foto-foto hasil dan proses pekerjaan yang dilakukan dapat ditampilkan pula di bagian ini. Pastikan penjelasan yang anda cantumkan meliputi proses perancangan dari awal hingga akhir.

3.3.1.1 Tahap 1: Data Understanding

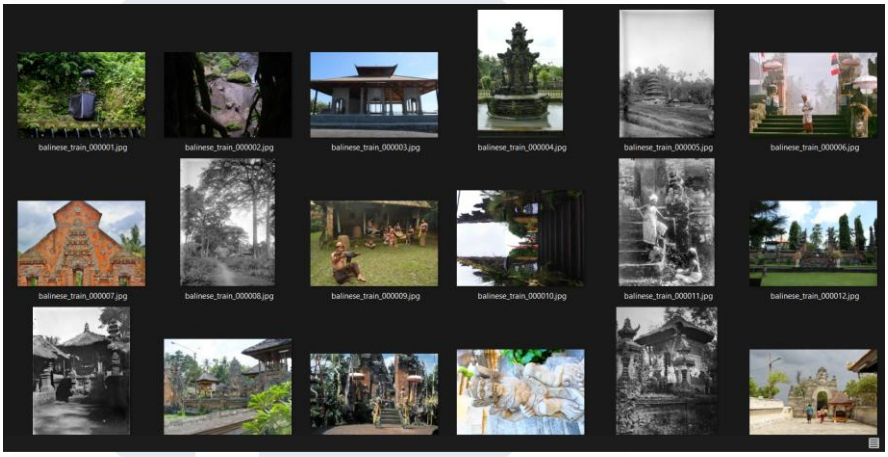
Tahapan ini merupakan tahapan awal dalam melakukan pembangunan model. Proses *data understanding* ini adalah memahami *dataset* yang diberikan oleh pihak panitia LOGIKA UI 2025 dari *kaggle.com*. Dataset yang diberikan berupa gambar yang berukuran 4,23 GB yang dibagi menjadi 2 folder yaitu folder *train* dan *test*. Pada bagian folder train terdapat 5 subfolder yang masing masing berisikan gambar dari setiap budaya di Indonesia. Subfolder yang diberikan ada 5 yaitu berupa folder *Balinese* yang memiliki 776 gambar, lalu folder batak yang memiliki 95 gambar, folder dayak yang memiliki 69 gambar, folder *javanese* yang memiliki 249 gambar, dan folder minangkabau yang memiliki 563 gambar. Total gambar pada folder *train* sebanyak 1.752 gambar . Lalu untuk folder *test* berisikan 444 gambar dengan budaya di Indonesia dari 5 suku yang di acak.



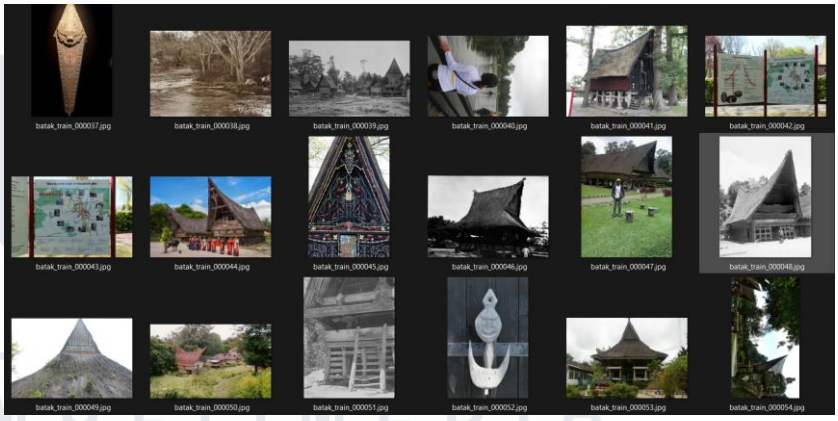
Gambar 3. 3 Hasil Gambar per folder

Berdasarkan gambar 3.3 diatas yang merupakan hasil dari total gambar per folder, sehingga mengetahui apakah ada kelas yang jumlah gambarnya lebih sedikit dibanding yang lainnya. Dan dari hasil yang didapatkan bahwa gambar untuk adat batak dan dayak bisa dibilang lebih sedikit dibanding ketiga suku lainnya yang berada di angka ratusan.

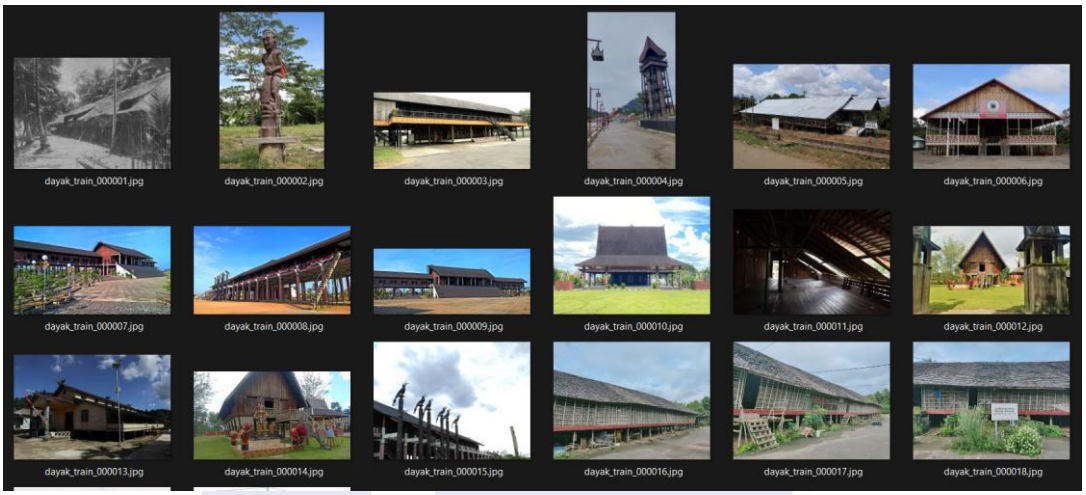
Berikut merupakan contoh dari gambar setiap kategori budaya yang diberikan dari *dataset train* dan contoh gambar *dataset test*:



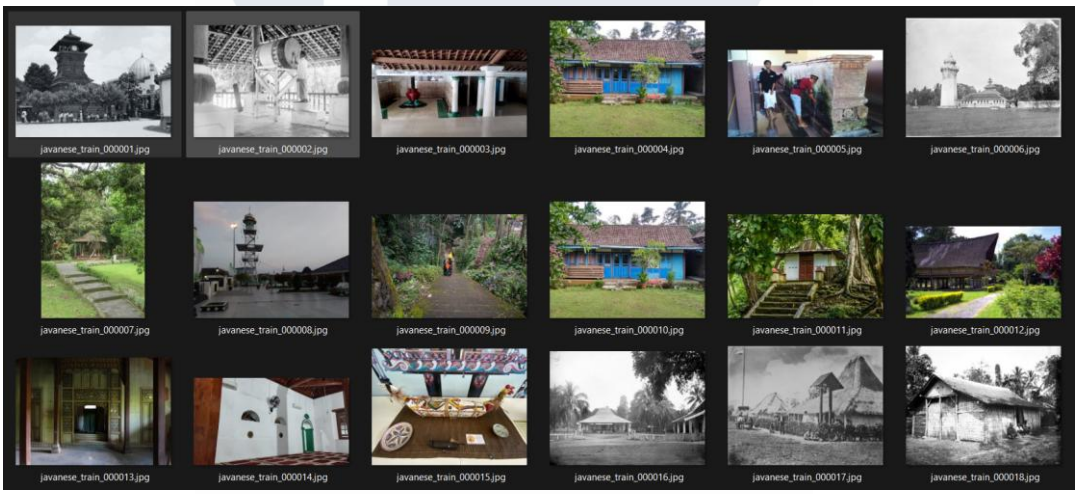
Gambar 3. 4 Contoh Dataset Train Balinese



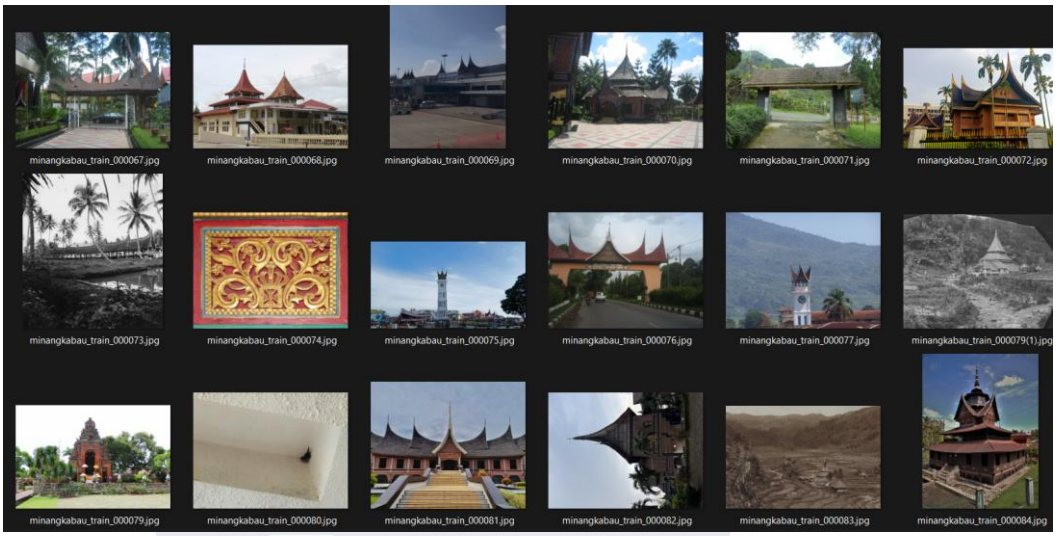
Gambar 3. 5 Contoh Dataset Train Batak



Gambar 3. 6 Contoh Dataset Train Dayak



Gambar 3. 7 Contoh Dataset Train Javanese



Gambar 3. 8 Contoh Dataset Train Minangkabau

Gambar 3.4, gambar 3.5, gambar 3.6, gambar 3.7 dan gambar 3.8 merupakan salah satu dari contoh gambar dari dataset train untuk budaya dari setiap adat mulai dari balinese, batak, dayak, javanese, dan minangkabau yang digunakan untuk input pada proses pelatihan model klasifikasi gambar budaya di Indonesia. Penyertaan contoh gambar ini diharapkan sehingga pembaca dapat menggambarkan karakteristik data gambar yang dipelajari oleh model.

Gambar 3. 9 Contoh Dataset Test

```
#Jumlah gambar di folder dataset

import os

file_test = os.listdir('./Test/Test/')

train_bali = os.listdir('./Train/Train/balinese')
train_batak = os.listdir('./Train/Train/batak')
train_dayak = os.listdir('./Train/Train/dayak')
train_java = os.listdir('./Train/Train/javanese')
train_minang = os.listdir('./Train/Train/minangkabau')

jumlah_test = len(file_test)
print("Gambar Test:", jumlah_test)

jumlah_bali = len(train_bali)
print("Gambar Train Bali:", jumlah_bali)

jumlah_batak = len(train_batak)
print("Gambar Train Batak:", jumlah_batak)

jumlah_dayak = len(train_dayak)
print("Gambar Train Dayak:", jumlah_dayak)

jumlah_java = len(train_java)
print("Gambar Train Java:", jumlah_java)


jumlah_minang = len(train_minang)
print("Gambar Train Minang:", jumlah_minang)
```

Gambar 3. 10 Kode untuk menghitung jumlah gambar di setiap folder dataset

Gambar 3.10 diatas menunjukan kode yang digunakan untuk menghitung jumlah gambar di setiap folder dataset dari folder train dan test. Import os pada kode ini digunakan sebagai perintah untuk dapat mengelola file dapat juga sebagai jembatan antara python dengan OS untuk dapat berinteraksi. Terdapat variabel – variabel yang didefinisikan untuk dapat mengelola file melalui directory – directory yang diberikan seperti : file_test = os.listdir(‘./Test/Test/’) perintah ini digunakan untuk dapat mengelola file dari folder test, train_bali = os.listdir(‘/Train/Train/balinese’) perintah ini digunakan untuk dapat mengelola file dari folder bali yang terdapat didalam folder train, train_batak = os.listdir(‘/Train/Train/batak’) perintah ini digunakan untuk dapat mengelola file dari folder batak yang terdapat

didalam folder train, train_dayak = os.listdir('/Train/Train/dayak') perintah ini digunakan untuk dapat mengelola file dari folder dayak yang terdapat didalam folder train, train_java = os.listdir('/Train/Train/java') perintah ini digunakan untuk dapat mengelola file dari folder java yang terdapat didalam folder train, train_minang = os.listdir('/Train/Train/minangkabau') perintah ini digunakan untuk dapat mengelola file dari folder minangkabau yang terdapat didalam folder train. Terdapat fungsi len() ataupun length() yang digunakan untuk menghitung jumlah file dari variabel yang telah dideklarasikan seperti :len(file_test) perintah ini digunakan untuk menghitung jumlah file dari folder test, len(jumlah_bali) perintah ini digunakan untuk menghitung jumlah file dari folder bali, len(jumlah_batak) perintah ini digunakan untuk menghitung jumlah file dari folder batak, len(jumlah_dayak) perintah ini digunakan untuk menghitung jumlah file dari folder dayak, len(jumlah_java) perintah ini digunakan untuk menghitung jumlah file dari folder java, len(jumlah_minang) perintah ini digunakan untuk menghitung jumlah file dari folder minang.

3.3.1.2 Tahap 2: Data Preprocessing

Tahapan ini menjadi tahapan kedua dalam pembangunan model. Setelah melalui tahapan data understanding kemudian data akan diolah agar data lebih terstruktur dan memiliki format yang sama pada setiap gambar di setiap folder nya. Tahapan ini diperlukan agar memiliki format yang sama agar memudahkan saat pembangunan model saat data digunakan, selain tahapan data preprocessing perlu dilakukan karna memiliki pengaruh terhadap performa model saat melakukan training dan akan mempengaruhi nilai akurasi yang didapatkan saat model selesai melakukan training.

```
train_transform = transforms.Compose([
    transforms.RandomResizedCrop(IMG_SIZE),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.RandomPerspective(distortion_scale=0.2, p=0.3),
    transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3),
    transforms.RandomGrayscale(p=0.1),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225]),
])

test_transform = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225])
])

train_dataset = datasets.ImageFolder("./Train/Train", transform=train_transform)
train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
```

Gambar 3. 11 Kode untuk memotong rasio gambar

Pada **g**ambar 3.11 kode ini menunjukkan tahapan data preprocessing Terdapat variabel train_transform yang memiliki beberapa perintah seperti RandomResizeCrop yang digunakan untuk memotong rasio secara acak pada gambar dengan ukuran 224x224, RandomHorizontalFlip digunakan untuk memutar agar gambar secara acak menjadi horizontal, RandomRotation digunakan untuk memutar gambar secara acak kurang lebih 15 derajat, RandomPerspective digunakan agar seolah – olah dipandang dengan cara pandang lain atau disebut dengan distoris perspektif, ColorJitter digunakan untuk mengatur kecerahan, kontras, dan saturasi atau seberapa kuat warna pada gambar namun perintah ini dilakukan secara acak, RandomGrayScale digunakan untuk mengubah warna memiliki channel warna yang sama yaitu GrayScale yang awalnya gambar memiliki channel warna RGB namun perintah ini dilakukan secara acak dan hanya 10% yang diubah dari keseluruhan gambar yang digunakan, ToTensor digunakan untuk mengubah format pada gambar yang dipahami oleh pytorch seperti mengubah konversi gambar yang

awalnya HWC atau height, width, channel diubah menjadi CHW atau channel, height,width, Normalize untuk menormalisasi gambar dengan menggunakan nilai yang mean dan nilai standard deviation atau dapat menggunakan nilai default model. Selain variabel train_transform terdapat juga variabel test_transform yang memiliki beberapa perintah seperti Resize digunakan untuk memotong rasio gambar, ToTensor digunakan untuk mengubah format pada gambar yang dipahami oleh pytorch seperti mengubah konversi gambar yang awalnya HWC atau height, width, channel diubah menjadi CHW atau channel, height,width, Normalize untuk menormalisasi gambar dengan menggunakan nilai yang mean dan nilai standard deviation atau dapat menggunakan nilai default model.

Pada tahapan data preprocessing terdapat beberapa tahapan seperti transform namun juga ada tahapan balancing data yang digunakan agar data yang awalnya tidak seimbang atau dapat diartikan sebagai salah satu kategori terlalu mendominasi kemudian dengan balancing data distribusi atau penyebaran data menjadi seimbang. Tahapan ini penting dilakukan agar model tidak hanya belajar pada salah satu kategori yang telalu mendominasi.

```python
from torch.utils.data import DataLoader, WeightedRandomSampler
from collections import Counter
import torch

# hitung jumlah gambar per kelas
class_counts = Counter([label for _, label in train_dataset.samples])
print("Class counts:", class_counts)

# total jumlah kelas
num_classes = len(class_counts)

# buat weight (semakin kecil jumlah data, semakin besar weight)
class_weights = {cls: 1.0 / count for cls, count in class_counts.items()}

# buat list sample_weights untuk tiap data
sample_weights = [class_weights[label] for _, label in train_dataset.samples]
sample_weights = torch.DoubleTensor(sample_weights)

# definisikan WeightedRandomSampler
sampler = WeightedRandomSampler(
    weights=sample_weights,
    num_samples=len(sample_weights),  # jumlah total sample yg akan diambil
    replacement=True  # penting supaya bisa oversample
)

# buat DataLoader dengan sampler
train_loader = DataLoader(
    train_dataset,
    batch_size=32,
    sampler=sampler,
    generator=torch.Generator().manual_seed(42)
)
```

Gambar 3. 12 Kode untuk melakukan balancing data

Berdasarkan gambar 3.12 diatas menunjukkan yang digunakan untuk melakukan tahapan balancing data. Dengan langkah pertama yaitu menampilkan jumlah gambar pada setiap kategori atau class seperti pada kode berikut class_counts = Counter([label for _, label in train_dataset.samples]) kemudian hasil dari perintah tersebut dengan menggunakan print("Class counts:", class_counts) yang digunakan untuk menampilkan output dari setiap jumlah gambar pada setiap kategori atau class. Output dari perintah tersebut adalah pada class 0 yaitu bali memiliki jumlah gambar 776 gambar, class 1 yaitu batak memiliki jumlah gambar 95 gambar, class 2 yaitu dayak memiliki jumlah gambar 69 gambar, class 3 yaitu java memiliki jumlah gambar 249 gambar, class 4 yaitu

minangkabau memiliki jumlah gambar 563 gambar. Setelah tahapan yang telah dilakukan kemudian tahapan selanjutnya adalah balancing data yang dapat diartikan bahwa kategori yang memiliki bobot ataupun jumlah data yang rendah maka bobot akan ditambah supaya data menjadi seimbang seperti pada pengguna kode class_weights = {cls: 1.0 / count for cls, count in class_counts.items()}. Setelah melakukan balancing data maka data akan seimbang seperti berikut: class 0 yaitu bali memiliki jumlah gambar 349 gambar, class 1 yaitu batak memiliki jumlah gambar 357 gambar, class 2 yaitu dayak memiliki jumlah gambar 350 gambar, class 3 yaitu java memiliki jumlah gambar 359 gambar, class 4 yaitu minangkabau memiliki jumlah gambar 337 gambar.

3.3.1.3 Tahap 3: Modelling

Tahapan ini menjadi tahapan utama saat pembangunan model yaitu tahapan modelling, tahapan ini sebagai aspek penting yang harus dilakukan untuk dapat membangun model agar dapat melakukan klasifikasi gambar sesuai dengan kategori asal budaya di Indonesia. Untuk model awal mengunakan efficientnet_b1 yang tersedia pada modul timm yang telah dilakukan klasifikasi gambar dengan dataset gambar yang besar. Maka dapat disebut bahwa efficientnet_b1 merupakan model yang siap dipakai untuk klasifikasi gambar bukan model yang harus dibuat dari awal atau dapat disebut dengan scratch, meskipun itu model ini dapat dilakukan disesuaikan untuk dapat disesuaikan dengan dataaset yang digunakan seperti pada variabel model.classifier yang terdapat penyesuaian seperti : nn.Linear(num_features, 512) yang digunakan untuk mengubah layer FC yang awalnya berupa backbone menjadi 512 neuron, nn.ReLU() yang digunakan untuk mengaktivasi ReLU agar model dapat belajar dengan lebih kompleks, nn.Dropout() yang digunakan untuk mengurangi terjadinya overfitting, dan nn.Linear() yang digunakan untuk menghasilkan output sesuai jumlah kategori yang terdapat pada dataset.

```python
import timm
import torch.optim as optim

effnet = timm.create_model("efficientnet_b1", pretrained=True, num_classes=0)
for param in effnet.parameters():
    param.requires_grad = False  # freeze layer

# DeiT-Small sebagai backbone utama
deit = timm.create_model("deit_small_patch16_224", pretrained=True, num_classes=0)

class HybridDeiT_EfficientNet(nn.Module):
    def __init__(self, deit_model, eff_model, num_classes, dropout=0.4):
        super().__init__()
        self.deit = deit_model
        self.eff = eff_model
        self.dropout = nn.Dropout(dropout)
        self.fc = nn.Linear(self.deit.num_features + self.eff.num_features, num_classes)

    def forward(self, x):
        feat_deit = self.deit(x)  # (B, 384)
        feat_eff = self.eff(x)    # (B, 1280)
        fused = torch.cat([feat_deit, feat_eff], dim=1)  # concat
        fused = self.dropout(fused)
        out = self.fc(fused)
        return out

model = HybridDeiT_EfficientNet(deit, effnet, num_classes=num_classes)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

Gambar 3. 13 Kode untuk menjalankan model 2

Berdasarkan gambar 3.13 diatas yang menunjukkan kode yang digunakan untuk model ke – 2, tahapan ini dilakukan untuk dapat meningkatkan akurasi setelah model dilatih menggunakan arsitektur ke –1. Model ke – 2 menggunakan DeiT dengan memanggil modul yang terdapat pada timm seperti berikut: deit = timm.create_model("deit_small_patch16_224", pretrained=True, num_classes=0) dengan kode ini maka model DeiT yang terdapat pada modul timm dipanggil tanpa menggunakan classifier bawaan maka model hanya akan mengembalikan fitur (feature vector). Kemudian terdapat class yang digunakan untuk menggabungkan kedua model yaitu efficientnet_b1 dengan DeiT menjadi hybrid arsitektur, dengan teknik dapat meningkatkan akurasi model untuk melakukan klasifikasi gambar.

3.3.1.4 Tahap 4: Training

Tahapan ini merupakan tahapan yang dilakukan setelah tahapan membangun model sudah dilakukan. Tahapan training merupakan tahapan

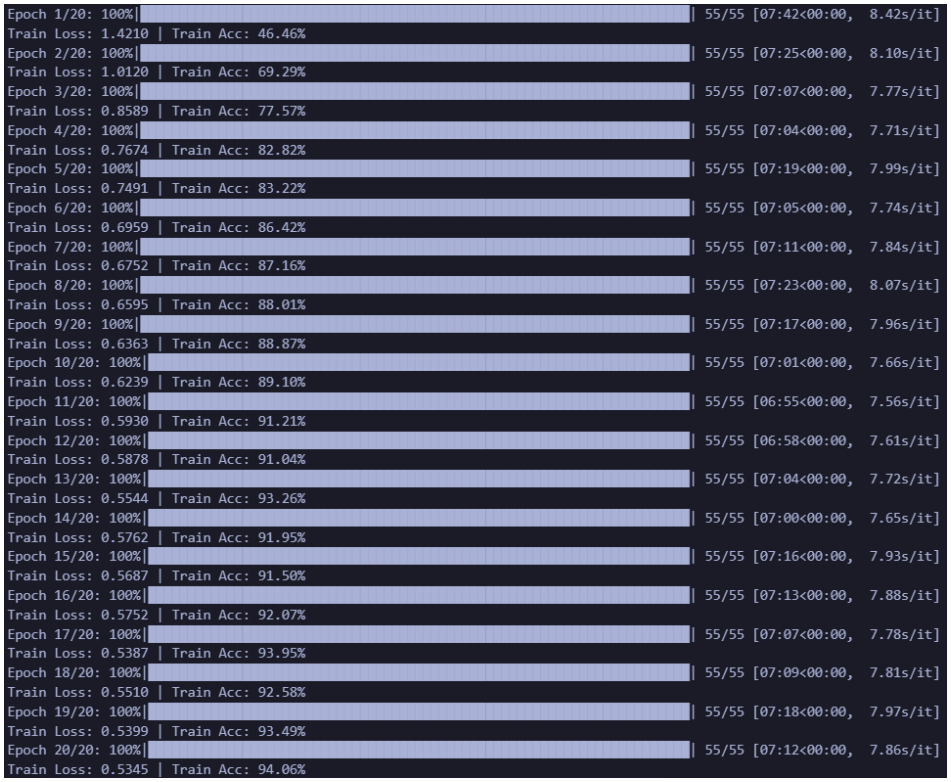
yang penting dilakukan dimana model akan belajar sesuai dengan struktur model yang telah disesuaikan dengan dataset. Dalam tahapan ini model akan belajar pada setiap epoch melalui pengulangan pada epoch, model akan mempelajari fitur ataupun pola. Model akan belajar sesuai epoch yang dijalankan agar model belajar secara lebih detail dan model bisa menghafal obyek pada gambar yang diberikan. Tujuan dari melakukan training adalah model dapat menghafal ataupun dapat melakukan klasifikasi gambar berdasarkan obyek yang diberikan dan mengetahui kategori yang sesuai dari obyek yang diberikan.

```python
for epoch in range(EPOCHS):
    model.train()
    running_loss, correct, total = 0.0, 0, 0

    for images, labels in tqdm(train_loader, desc=f"Epoch {epoch+1}/{EPOCHS}"):
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = outputs.max(1)
        total += labels.size(0)
        correct += predicted.eq(labels).sum().item()

    train_acc = 100. * correct / total
    train_loss = running_loss / len(train_loader)
    print(f"Train Loss: {train_loss:.4f} | Train Acc: {train_acc:.2f}%")

    scheduler.step()
```

Gambar 3. 14 Kode untuk menjalankan pelatihan model 1

Berdasarkan gambar 3.14 diatas yang menunjukkan bahwa terdapat kode pada model yang diulang sesuai dengan epoch yang telah ditentukan kemudian gambar akan dipindahkan pada GPU untuk dapat dikomputasi lebih cepat, terdapat penggunaan optimizer.step() yang digunakan untuk mengaktifkan penggunaan optimizer yang ditentukan. Penggunaan train_acc = 100 * correct / total yang digunakan untuk menghitung akurasi yang kinerja model dari setiap epoch, kemudian train_loss = running_loss

/ len(train_loader) digunakan untuk menghitung loss dari setiap epoch.



Gambar 3. 15 Gambar 3.15 Hasil tahapan training model 1

Berdasarkan gambar 3.15 diatas yang menunjukkan tahapan training pada model – 1 selama 20 epoch :

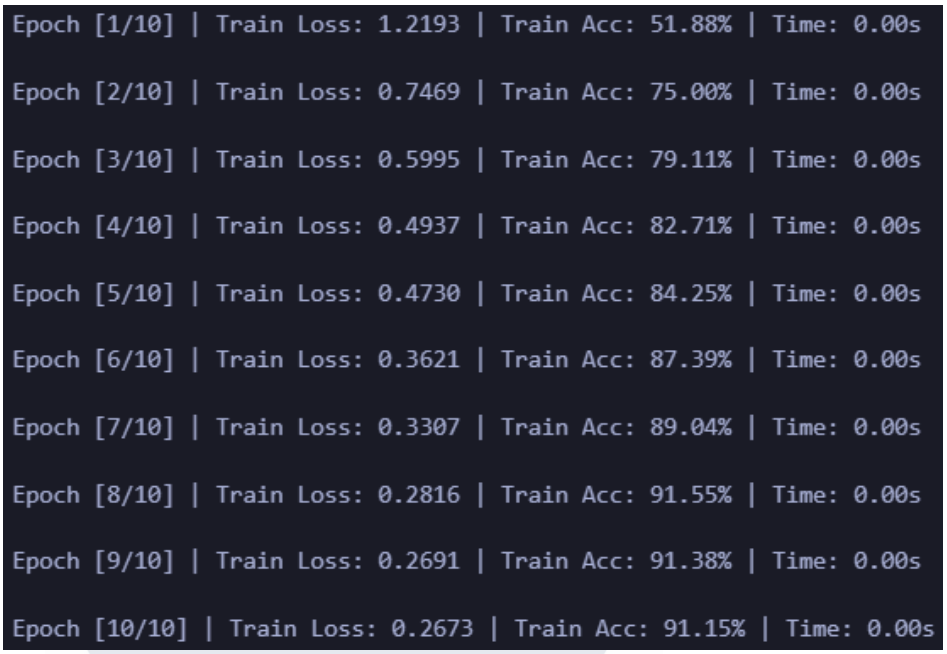
1. Epoch 1 memiliki train akurasi 46.46% dan train loss 1.4210.
2. Epoch 5 memiliki train akurasi 83.22% dan train loss 0.7491.
3. Epoch 10 memiliki train akurasi 89.10% dan train loss 0.6239.
4. Epoch 15 memiliki train akurasi 91.50% dan train loss 0.5687.
5. Epoch 20 memiliki train akurasi 94.06% dan train loss 0.5345.

Yang dijalankan dengan 20 epoch yang menunjukkan bahwa model belajar dengan baik yang dapat dillihat dari peningkatan pada akurasi yang didapatkan yang meningkat dengan stabil, tidak adanya peningkatan yang terlalu signifikan, kemudian terdapat penurunan pada train loss dikarenakan penggunaan optimizer yang berjalan dengan baik.

```python
import time

start_time = time.time()
def train_one_epoch(model, loader, optimizer, criterion):
    model.train()
    running_loss, correct, total = 0.0, 0, 0
    for imgs, labels in tqdm(loader, desc="Training", leave=False):
        imgs, labels = imgs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(imgs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, preds = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (preds == labels).sum().item()

    train_acc = 100 * correct / total
    return running_loss / len(loader), acc

end_time = time.time()
epoch_time = end_time - start_time


num_epochs = 10
best_acc = 0

for epoch in range(num_epochs):
    train_loss, train_acc = train_one_epoch(model, train_loader, optimizer, criterion)
    scheduler.step()

    print(f"Epoch [{epoch+1}/{num_epochs}] | Train Loss: {train_loss:.4f} | "
          f"Train Acc: {train_acc:.2f}% | "
          f"Time: {epoch_time:.2f}s")
```

Gambar 3. 16 Kode untuk menjalankan pelatihan model 2

Berdasarkan gambar 3.16 menujukkan bahwa terdapat kode penggunaan def train_one_epoch yang digunakan untuk mengaktifkan beberapa perintah seperti forward pass, backward pass, menghitung loss dan optimizer.step(). Fungsi penggunaan dari optimizer.step() digunakan untuk memperbarui pada bobot model. Pada model _train() yang akan diulang sesuai epoch yang ditentukan kemudian setiap gambar akan dipindahkan ke GPU yang akan dikomputasi lebih cepat. Kemudian terdapat train_acc = 100 * correct / total yang digunakan untuk menghitung akurasi yang kinerja model dari setiap epoch.



Gambar 3. 17 Hasil training mode 2

Berdasarkan gambar 3.17 diatas yang menunjukkan tahapan training pada model – 2 selama 10 epoch :

1. Epoch 1 memiliki train akurasi 51.88% dan train loss 1.2193
2. Epoch 5 memiliki train akurasi 84.25% dan train loss 0.4730
3. Epoch 10 memiliki train akurasi 91.15% dan train loss 0.2673.

Yang dijalankan dengan 10 epoch yang menunjukkan bahwa model belajar dengan baik dapat dilihat dengan peningkatan akurasi yang stabil dan tidak adanya peningkatan yang terlalu signifikan dan memiliki train loss yang terus menurun pada setiap epoch nya.

3.3.1.5 Tahap 5: Evaluation

Tahapan evaluasi menjadi tahapan terakhir pada tahapan pembangunan model. Tahapan ini digunakan untuk mengetahui performa atau kemampuan model dalam melakukan klasifikasi gambar dengan model yang telah dibangun. Dengan mengetahui kekurangan dalam model saat belajar, kemudian model dapat ditingkatkan sesuai dengan dengan kekurangan yang ditemukan pada tahapan evaluasi. Secara sistematis

tahapan ini menggunakan classification report dengan menyajikan laporan dalam bentuk tabel yang berisikan :

1. Class merupakan kategori yang telah diprediksi oleh model.
2. Precision merupakan metrik yang menjadi ukuran keakuratan model terhadap kategori.
3. Recall merupakan mengukur seberapa banyak data positif yang berhasil ditemukan oleh model dari seluruh data positif yang sebenarnya ada.
4. F1- Score merupakan nilai rata – rata dari nilai presisi dan recall.
5. Support merupakan jumlah kemunculan kategori saat tahapan evaluasi.

	precision	recall	f1-score	support
balinese	0.943	0.940	0.941	316
batak	0.976	0.957	0.967	347
dayak	0.987	0.995	0.991	373
javanese	0.935	0.940	0.938	367
minangkabau	0.932	0.940	0.936	349
accuracy			0.955	1752
macro avg	0.955	0.954	0.954	1752
weighted avg	0.955	0.955	0.955	1752

Gambar 3. 18 Hasil classification report untuk model 1

Berdasarkan gambar 3.18 diatas yang menunjukkan *classificaition report* pada model – 1, berikut merupakan rincian detail dari nilai presisi yang didapatkan di 0.93 – 0.98 dari setiap kategori seperti :

1. Dayak memiliki nilai presisi yang tertinggi dengan nilai 0.987
2. Batak yang menjadi kategori kedua yang memiliki nilai presisi dengan nilai 0.976
3. Bali yang memiliki nilai presisi dengan nilai 0.943

4. Java yang memiliki nilai presisi dengan nilai 0.935
5. Minangkabau yang memiliki nilai presisi dengan nilai 0.932.

Nilai recall yang didapatkan dapat diartikan bahwa model dapat mengklasifikasikan gambar dengan cukup baik, berikut merupakan rincian detail dari nilai recall yang didapatkan di 0.94 – 0.99, dari setiap kategori seperti :

1. Dayak yang memiliki nilai recall dengan nilai 0.995.
2. Batak yang memiliki nilai recall dengan nilai 0.957.
3. Bali yang memiliki nilai recall dengan nilai 0.940.
4. Java yang memiliki nilai recall dengan nilai 0.940.
5. Minangkabau yang memiliki nilai recall dengan nilai 0.940.

Pada f1-score mendapatkan nilai di kisaran 0.93 sampai 0.99 yang dapat dikatakan bahwa penyebaran ataupun distribusi data cukup baik, dikarenakan tidak adanya data yang dianggap mayoritas atau tidak , berikut merupakan rincian detail dari nilai f1-score yang didapatkan di 0.93 – 0.99, dari setiap kategori seperti :

1. Dayak yang memiliki nilai f1-score dengan nilai 0.991.
2. Batak yang memiliki nilai f1-score dengan nilai 0.967.
3. Bali yang memiliki nilai f1-score dengan nilai 0.941.
4. Java yang memiliki nilai f1-score dengan nilai 0.938.
5. Minangkabau yang memiliki nilai f1-score dengan nilai 0.936.

	precision	recall	f1-score	support
balinese	0.863	0.902	0.882	336
batak	0.960	0.936	0.948	357
dayak	0.970	0.982	0.976	333
javanese	0.894	0.906	0.900	363
minangkabau	0.897	0.860	0.878	363
accuracy			0.916	1752
macro avg	0.917	0.917	0.917	1752
weighted avg	0.917	0.916	0.916	1752

Gambar 3. 19 Hasil classification report untuk model 2

Berdasarkan gambar 3.19 diatas yang menunjukkan classification report pada model – 2 berikut merupakan rincian detail dari nilai presisi yang didapatkan di 0.86 – 0.97, berikut merupakan rincian detail nilai presisi yang didapatkan dari setiap kategori di sebagai berikut :

1. Dayak memiliki nilai presisi yang tertinggi dengan nilai 0.970.
2. Batak menjadi kategori kedua yang memiliki nilai presisi yang tinggi dengan nilai 0.960.
3. Minangkabau yang memiliki nilai presisi dengan nilai 0.897.
4. Java yang memiliki nilai presisi dengan nilai 0.894.
5. Bali yang memiliki nilai presisi dengan nilai 0.863.

Nilai recall yang didapatkan dapat diartikan bahwa model dapat mengklasifikasikan gambar dengan cukup baik dengan nilai yang didapatkan di 0.86 – 0.98 berikut merupakan rincian detail nilai recall yang didapatkan dari setiap kategori seperti berikut :

1. Dayak yang memiliki nilai recall dengan nilai 0.982.
2. Batak yang memiliki nilai recall dengan nilai 0.936.
3. Java yang memiliki nilai recall dengan nilai 0.906.

4. Bali yang memiliki nilai recall dengan nilai 0.902.
5. Minangkabau yang memiliki nilai recall dengan nilai 0.860.

Pada f1-score mendapatkan nilai di kisaran 0.87 sampai 0.97 yang dapat dikatakan bahwa distribusi data yang cukup baik tidak adanya data yang terlalu mendominasi, berikut merupakan rincian detail nilai f1-score yang didapatkan dari setiap kategori seperti berikut :

1. Dayak yang memiliki nilai f1-score dengan nilai 0.976.
2. Batak yang memiliki nilai f1-score dengan nilai 0.948.
3. Java yang memiliki nilai f1-score dengan nilai 0.900.
4. Bali yang memiliki nilai f1-score dengan nilai 0.882.
5. Minangkabau yang memiliki nilai f1-score dengan nilai 0.878.

3.3.2 Kendala yang Ditemukan

Selama melaksanakan PRO STEP: Road to Champion ada beberapa kendala yang ditemukan dalam pengerjaan kompetisi:

1. Kesulitan dalam mencari studi kasus yang mempunyai persamaan dengan studi kasus yang diberikan oleh panitia kompetisi, sehingga terlambat dalam memahami cara dan teknis pengerjaan soal yang diberikan.
2. Kesulitan ketika ingin mengerjakan kompetisi karena kompetisi yang diikuti tidak sesuai dengan peminatan yang dipilih ketika masa perkuliahan.
3. Kesulitan ketika mengerjakan kompetisi dan menjalankan code pada Google Collab yang bergantung pada internet dan memiliki keterbatasan pada kinerja GPU, dan internet yang digunakan sedang dalam kondisi yang tidak stabil menyebabkan code berhenti berjalan pada saat menjalankan proses training.

4. Kurangnya informasi yang detail terhadap teknis pengumpulan hasil dari pengerjaan kompetisi LOGIKA UI 2025 melalui Kaggle Competition.

3.3.3 Solusi atas Kendala yang Ditemukan

Solusi yang dilakukan untuk mengatasi kendala-kendala yang dihadapi selama pengerjaan kompetisi PRO STEP: Road to Champion:

1. Mencari studi kasus dari internet yang memiliki model image classification yang akan di pakai dan bagus sesuai dengan ketentuan gambar gambar yang di dapatkan.
2. Mengulang dan memahami kembali materi terkait data science yang pernah di pelajari selama masa perkuliahan, untuk mempersiapkan diri dalam melaksanakan kompetisi.
3. Berhenti menggunakan Google Collab dan mendiskusikannya untuk mengganti tools dengan Jupyter Notebook dan setiap anggota menjalankan kode dengan model yang berbeda beda sehingga lebih mempercepat mendapatkan akurasi dari berbagai model.
4. Mencari tahu sendiri teknis pengumpulan file ketika mengikuti Kaggle Competition dari internet.

3.4 Hasil Lomba/Kompetisi

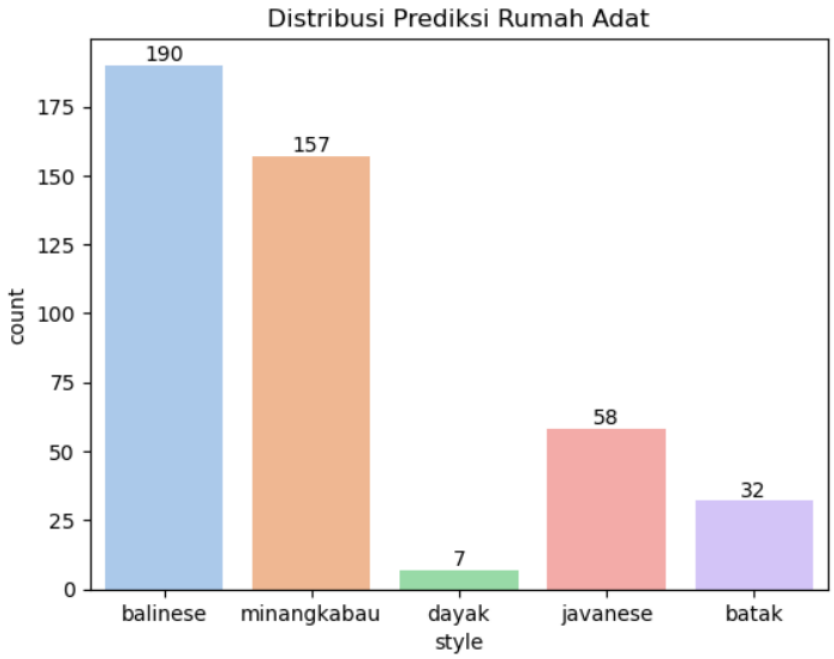
Setelah menyelesaikan pengerjaan kompetisi LOGIKA UI 2025 data science competition, dengan pembuatan dari berbagai model image classification rumat adat yang ada di Indonesia berhasil mendapatkan tingkat akurasi yang tinggi namun ketika melakukan pengumpulan hasil pengerjaan kompetisi LOGIKA UI 2025 di kaggle.com hasil yang didapatkan kurang memuaskan. Kelompok berusaha mencari cara untuk meningkatkan tingkat akurasi namun setelah merubah dan menambahkan tahapan dari mulai *data preprocessing* sampai ke *evaluation* hasil

yang didapatkan tidak berpengaruh secara signifikan karena hasil yang didapatkan tidak terlalu berbeda dengan hasil yang pertama. Hasil yang didapatkan dari pengerjaan adalah file csv dengan prediksi disetiap gambar pada *folder test*. Berikut merupakan contoh 10 prediksi teratas untuk data test yang dihasilkan

	id	style
0	Test_001	balinese
1	Test_002	minangkabau
2	Test_003	minangkabau
3	Test_004	balinese
4	Test_005	balinese
5	Test_006	dayak
6	Test_007	balinese
7	Test_008	balinese
8	Test_009	javanese
9	Test_010	minangkabau

Gambar 3. 20 Hasil Submisi dengan 10 Prediksi

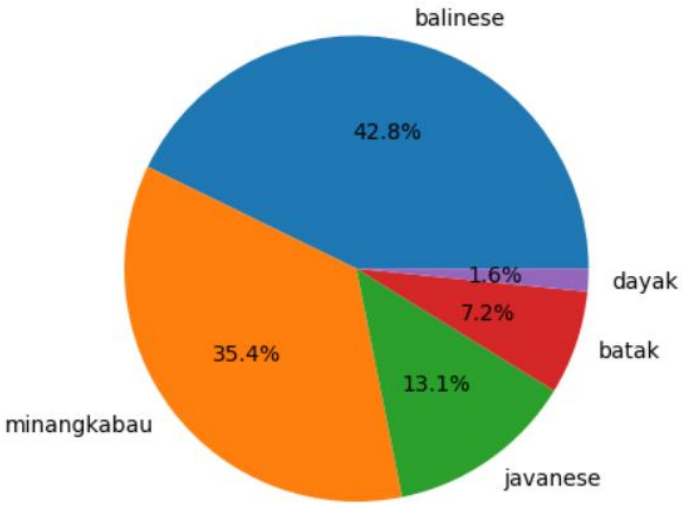
Gambar 3.20 diatas menampilkan sepuluh prediksi teratas yang dihasilkan model pada data gambar dari folder test, yang digenerate dalam format .csv sebagaimana ditentukan oleh panitia kompetisi. Visualisasi ini menggambarkan keluaran langsung dari model EfficientNet-B1 setelah melalui tahapan *data preprocessing, modeling, training, dan evaluation*. Setiap baris dalam hasil tersebut menunjukkan label budaya di Indonesia yang diprediksi beserta struktur penyusunan prediksi yang diwajibkan sesuai dengan *sample submission* yang diberikan untuk proses penilaian di *Kaggle.com*.



Gambar 3. 21 Visualisasi Distribusi Prediksi

Gambar 3.21 di atas menampilkan distribusi jumlah prediksi model untuk masing-masing kelas budaya di Indonesia pada dataset gambar pada folder test. Grafik menunjukkan bahwa model *EfficientNet-B1* memberikan prediksi terbanyak pada kelas *balinese* dengan total sebanyak 190 prediksi, lalu pada kelas minangkabau dengan total sebanyak 157 prediksi. Sementara itu, kelas *javanese* dan batak memperoleh prediksi yang lebih rendah, masing-masing total 58 prediksi dan 32 prediksi. Adapun kelas dayak menjadi kelas dengan total prediksi paling sedikit, yaitu hanya 7 prediksi. Berdasarkan visualisasi distribusi ini memberikan gambaran awal mengenai kecenderungan model dalam memetakan data test ke kelas tertentu.

Ketidakseimbangan dalam distribusi prediksi menunjukkan adanya kemungkinan bahwa model memiliki bias terhadap kelas yang memiliki ciri visual yang lebih kuat atau serupa dengan pola yang telah dipelajari saat pelatihan. Dominasi pada kelas balinese dan minangkabau bisa mengindikasikan bahwa model lebih yakin dalam mengenali karakteristik visual dari kedua kelas itu, sementara prediksi yang sangat sedikit untuk kelas dayak menunjukkan bahwa model kesulitan menemukan pola yang mewakili adat tersebut.



Gambar 3. 22 Visualisasi Persentase Prediksi

Gambar 3.22 diatas ini menampilkan distribusi prediksi dalam bentuk persentase oleh model untuk lima kategori rumah tradisional yang diikutsertakan dalam kompetisi. Dalam tampilan tersebut, tampak bahwa model menunjukkan prediksi terbesar untuk kategori balinese, yaitu 42,8%, diikuti kategori minangkabau yang mencapai 35,4%. Sementara kategori javanese mendapatkan 13,1% dari total prediksi, dan kategori batak mendapat 7,2%. Kategori dayak berada di urutan terakhir dengan prediksi terkecil, yaitu hanya 1,6%. Persentase ini memberikan pemahaman yang lebih jelas tentang kecenderungan dan pola dari prediksi model daripada grafik distribusi frekuensi sebelumnya. Visualisasi ini menunjukkan bahwa model tidak membagi hasil prediksi secara merata di semua kelas, tetapi lebih condong pada dua kelas utama, yaitu balinese dan minangkabau.

Ketidakseimbangan ini menunjukkan bahwa kemampuan model dalam mengenali ciri visual di setiap kelas belum seimbang. Sangat rendahnya proporsi di kelas dayak menunjukkan bahwa model belum bisa mengenali pola visual unik dari kelas ini dengan baik, sehingga kinerjanya pada kelas itu mungkin masih kurang.

Setelah melakukan submit model dengan akurasi testing sebesar 91,6%, hasil evaluasi di kaggle menunjukkan akurasi sebesar 75,8% yang masih belum sesuai dengan harapan. upaya optimalisasi kemudian dilakukan dengan mengganti arsitektur model menjadi resnet34 yang menghasilkan akurasi testing lebih tinggi, yaitu 96%, namun pada saat disubmisi ke kaggle justru memperoleh akurasi yang lebih rendah sebesar 64,7%, sehingga menunjukkan bahwa peningkatan akurasi testing tidak selalu berbanding lurus dengan performa pada data uji kompetisi. Berikut merupakan hasil 10 prediksi teratas dari optimalisasi model menggunakan ResNet34.

	id	style
0	Test_001	balinese
1	Test_002	minangkabau
2	Test_003	javanese
3	Test_004	javanese
4	Test_005	balinese
5	Test_006	dayak
6	Test_007	balinese
7	Test_008	dayak
8	Test_009	javanese
9	Test_010	minangkabau

Gambar 3. 23 Hasil Submisi 10 Prediksi dengan Model Optimalisasi

Tabel 3. 3 Tabel Perbandingan Model Utama dengan Model Optimalisasi

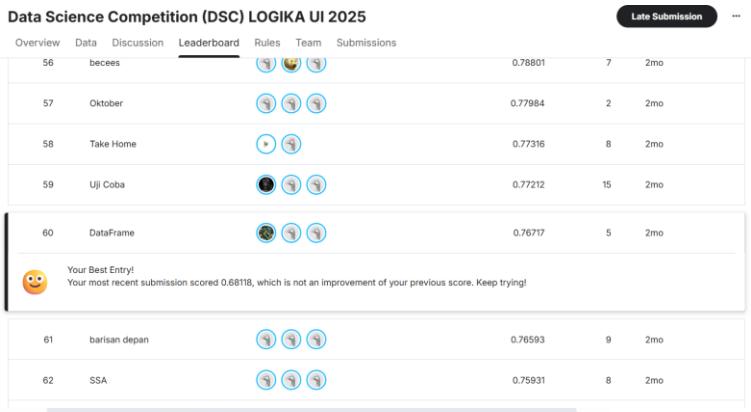
Aspek Evaluasi	EfficientNet-B1	ResNet34
Accuracy	91,6%	96,0%
Precision (macro avg)	91,7%	95,0%
Recall (macro avg)	91,7%	95,0%
F1-score (macro avg)	91,7%	95,0%
F1-score (weighted avg)	91,6%	96,0%

Berdasarkan tabel 3.3 diatas hasil *classification report* menunjukkan bahwa ResNet34 memiliki nilai precision, recall, dan f1-score yang lebih tinggi pada data evaluasi internal. namun, saat dilakukan pengujian melalui submisi di Kaggle, EfficientNet-B1 justru memberikan hasil yang lebih baik, yang menandakan kemampuan generalisasi yang lebih unggul pada data baru. hal ini menunjukkan bahwa performa tinggi pada evaluasi lokal tidak selalu mencerminkan hasil terbaik pada pengujian nyata, sehingga EfficientNet-B1 dipilih sebagai model utama.

Tabel 3. 4Tabel Perbandingan Model Utama dengan Model Optimalisasi

Aspek Perbandingan	EfficientNet-B1	ResNet 34
Kompleksitas model	Lebih Ringan	Lebih Kompleks
Stabilitas saat Traning	Stabil	Sangat Stabil
Akurasi Validasi	Lebih Rendah	Lebih Tinggi
Jumlah Parameter	Lebih sedikit	Lebih Banyak
Akurasi Testing	Lebih Tinggi	Lebih Rendah
Waktu Training	Lebih Efisien	Lebih Lama

Berdasarkan tabel 3.4 diatas menunjukkan bahwa resnet34 memiliki akurasi validasi dan stabilitas training yang lebih tinggi, namun dengan kompleksitas model, jumlah parameter, dan waktu training yang lebih besar. sementara itu, efficientnet-b1 lebih ringan, efisien, dan memberikan akurasi testing yang lebih baik, sehingga lebih unggul dalam hal generalisasi pada data baru. Dibawah berikut merupakan hasil dan peringkat yang didapatkan diakhir kompetisi sudah selesai.



Gambar 3. 23 Hasil Leaderboard Final

Berdasarkan gambar 3.24 diatas yang merupakan hasil peringkat akhir yang didapatkan setelah kompetisi selesai diselenggarakan adalah peringkat 60 dari 94 partisipan kelompok dengan total entry yang di submit ada di angka 5 kali. Dan terdapat peningkatan dari tingkat akurasi 0.76717 dan private score berada di 0.73474.

Gambar 3. 24 Sertifikat Kompetisi LOGIKA UI 2025

Gambar 3.25 diatas yang merupakan sertifikat kompetisi LOGIKA UI 2025 sebagai bukti keikutsertaan sebagai peserta dengan nomor peserta DSC0027 di *Data Science Competition* pada tinkgat nasional yang di selenggarakan pada tanggal 14 Juli 2025 sampai dengan 23 november 2025 secara daring dan luring. Dengan Mahasiswa dari Fakultas Matematika dan ilmu Pengetahuan Alam Universitas Indonesia sebagai panitia dan penyelenggara dari kompetisi ini.