BAB III

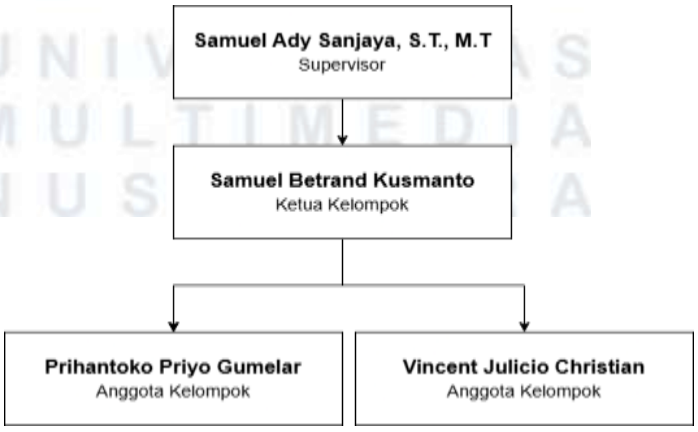
PELAKSANAAN PRO-STEP : ROAD TO CHAMPION

3.1 Kedudukan dan Koordinasi

Pelaksaanaan PRO – STEP : Road To Champion dilaksanakan secara berkelompok yang dibimbing oleh Bapak Samuel Ady Sanjaya, S.T., M.T. yang memberikan arahan serta pendapat untuk kelancaran dan keberhasilan dalam mengikuti kompetisi agar mendapatkan hasil yang memuaskan. Dalam pelaksanaan PRO-STEP : Road To Champion sebagai anggota tim yang berfokus dalam membantu tugas ketua tim seperti membantu menyusun strategi yang harus dilakukan dalam kompetisi dan mengimplementasikan strategi pada kompetisi agar dapat berjalan sesuai strategi yang telah disusun bersama ketua dan anggota tim.

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

   Berikut merupakan bagan alur yang menjelaskan kedudukan antara supervisor dan kelompok dalam pelaksanaan PRO – STEP : Road To Champion pada Gambar 3.1 Bagan Alur Koordinasi dan penjelasan tugas dari masing - masing kedudukan.



Gambar 3. 1 Bagan Alur Koordinasi

Penjelasan Tugas Struktur Kedudukan :

1. **Supervisor**

Dalam struktur kedudukan pada pelaksanaan PRO-STEP : Supervisor merupakan kedudukan tertinggi pada struktur ini yang merupakan posisi yang bertanggung jawab atas kinerja dari tim. Peran supervisor cukup penting dalam kelompok karena peran supervisor seperti pemberian arahan teknis maupun saran bagi seluruh anggota kelompok selama rangkaian kompetisi berlangsung. Selain itu peran supervisor bertugas memberikan evaluasi dari setiap tahapan yang telah dilakukan ataupun tahapan yang telah diselesaikan. Serta supervisor melakukan evaluasi terhadap hasil pengerjaan proyek yang telah dikerjakan oleh kelompok dan memberikan masukan terhadap konsep ataupun ide agar mendapatkan hasil maksimal pada kompetisi.

2. **Ketua Kelompok**

Sebagai pemimpin dalam kelompok ketua kompok Berperan untuk memastikan koordinasi dalam proses operasional dan komunikasi antar anggota berjalan dengan baik dan lancar untuk memastikan proses pengerjaan berjalan dengan baik. Dengan memberikan tugas dan tanggung jawab kepada anggota kelompok dengan jelas dan terstruktur agar proses tetap berjalan secara terstruktur. Selain itu ketua kelompok berperan dalam mengawasi progres pada setiap tahapan perkembangan setiap pengerjaan yang telah dilakukan dan memberikan evaluasi serta saran terhadap progres yang dilaporkan anggota kelompok.

3. **Anggota Kelompok**

Berperan untuk merealisasikan rencana ataupun tugas yang telah diberikan oleh ketua kelompok yang telah ditentukan dan

berperan aktif dalam diskusi bersama ketua kelompok selama pelaksanaan berlangsung. Selain itu anggota kelompok aktif dalam melaporkan hasil pengerjaan kepada ketua kelompok untuk mendapatkan evaluasi serta saran. Ketua kelompok memberikan tugas untuk mencoba model ResNet34 kemudian melakukan perbandingan model setelah dioptimasi dan sebelum dioptimasi.

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

Berikut pada Tabel 3.1 Detail Pekerjaan yang Dilakukan PRO-STEP : Road To Champion Program 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	Mendapatkan *brief* lomba dari penyelenggara lomba.	Mendapatkan *brief* lomba dari penyelenggara lomba melalui *technical meeting* yang telah dilakukan.
2	2	Diskusi terkait *brief* lomba dengan kelompok.	Melakukan diskusi awal dengan kelompok untuk memahami lebih lanjut terkait *brief* lomba yang diberikan.
3	3	*Brainstorming* dengan kelompok dan mengidentifikasi karakteristik dataset.	Identifikasi karakteristik untuk dataset yang digunakan dikarenakan dataset yang digunakan memiliki ukuran yang cukup besar dengan memiliki beragam karakteristik yang terdapat pada foto yang akan digunakan.

No.	Minggu	Proyek	Keterangan
4	4	Berdikusi untuk menentukan model dengan kelompok dan pembagian tugas	Mengidentifikasikan model yang sesuai dan relevan dengan dataset, yang akan diimplementasikan dengan kelompok dan dicoba untuk mendapatkan akurasi tinggi pada model, serta ketua kelompok membagikan tugas untuk dapat mencoba beberapa model secara mandiri.
5	5	Mengimplementasikan model yang telah ditentukan	Mencoba model secara mandiri yang telah ditentukan oleh Ketua Kelompok dan memilih model yang memiliki akurasi tertinggi.
6	6	Evaluasi terhadap model yang telah ditentukan	Diskusi dengan kelompok terkait hasil sementara dari beberapa eksperimen beberapa model yang telah ditentukan diawal.
7	7	Eksplorasi model lainnya untuk mendapat akurasi tertinggi	Melakukan eksperimen model lainnya sesuai dengan model yang telah ditentukan setelah melakukan evaluasi untuk dapat mendapatkan nilai akurasi yang terbaik, sebelum dilaporkan pada supervisor.
8	8	Melaporkan hasil pengerjaan kepada supervisor	Melaporkan hasil pengerjaan untuk mendapatkan masukan serta saran terhadap hasil pengerjaan.
9	9	Mengumpulkan hasil pengerjaan di Kaggle	Mengumpulkan hasil pengerjaan berupa file ipynb dan csv di halaman Kaggle Competition.

No.	Minggu	Proyek	Keterangan
10	10	Menyusun Bab I : Pendahuluan	Menyusun Bab I pada laporan yang mencakup latar belakang PRO-STEP : Road to Champion, tujuan dan manfaat PRO-STEP : Road to Champion dan deskripsi waktu dan prosedur PRO-STEP : Road to Champion.
11	11	Menyusun Bab II : Tentang Lomba / Kompetisi	Menyusun Bab II pada laporan yang mencakup dekripsi pelaksanaan lomba, alur pendaftaran lomba, portofolio hasil karya lomba dan output lomba yang dihasilkan.
12	12	Bimbingan dan menyusun Bab III pada sub-bab 3.1 hingga 3.2	Melakukan bimbingan terkait dengan pengerjaan Bab I dan Bab II yang telah dikerjakan sebelumnya dan menyusun Bab III pada sub-bab 3.1 terkait kedudukan dan koordinasi dan sub-bab 3.2 terkait pencatatan rangkuman mingguan proses pelaksanaan PRO-STEP : Road to Champion.
13	13	Menyusun Bab III pada sub-bab 3.3 dan 3.3.1	Menyusun Bab III pada sub-bab 3.3 terkait uraian pelaksanaan kerja, dan sub-bab 3.3.1 terkait proses pelaksanaan.
14	14	Menyusun Bab III pada sub-bab 3.3.1.1 hingga 3.3.1.5 dan dokumentasi	Menyusun Bab III pada sub-bab 3.3.1.1 hingga 3.3.1.5 yang merupakan sub-bab yang berisikan tahapan pengerjaan yang telah dilakukan dan melakukan dokumentasi dari setiap proses atau tahapan selama

No.	Minggu	Proyek	Keterangan
			membangun model.
15	15	Menyusun Bab IV : Penutup	Menyusun Bab IV terkait penutup laporan yang mencangkup kesimpulan dan saran bagi universitas dan bagi mahasiswa.
16	16	Bimbingan Final terkait laporan yang telah disusun	Melakukan bimbingan final terkait dengan laporan yang telah dikerjakan yang dilakukan oleh pembimbing.
17	17	Penyempurnaan akhir Laporan PRO-STEP : Road to Champion	Pengerjaan terhadap revisi yang diberikan saat melakukan bimbingan final dengan supervisor dan penyempurnaan akhir sehingga laporan telah sesuai dengan ketentuan.

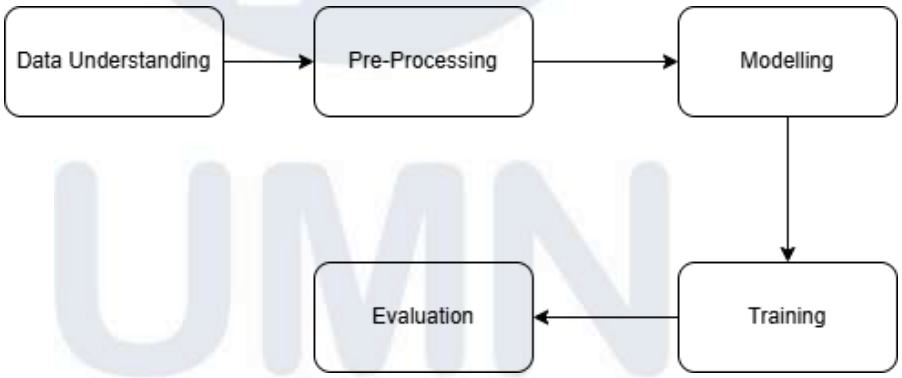
3.3 Uraian Pelaksanaan Kerja Dalam *PRO-STEP : Road To Champion Program*

Pelaksanaan kerja dalam PRO-STEP : Road to Champion dilakukan selama 17 minggu yang diawali dengan melakukan *brief* lomba yang diberikan kemudian melakukan pembahasan atau diskusi dengan kelompok terkait dengan *brief* lomba, kemudian melakukan brainstorming bersama kelompok yang membahas tentang karakteristik dataset yang digunakan pada saat membangun model. Kemudian ketua kelompok menentukan model yang akan digunakan dan membagi tugas pada anggota untuk melakukan eksperimen model secara mandiri untuk mengetahui akurasi model terhadap dataset. Sesuai dengan model yang telah ditentukan anggota melakukan implementasi model secara mandiri dan dilanjutkan melakukan evaluasi dengan ketua kelompok terkait hasil sementara yang didapatkan dari model yang telah ditentukan. Berdasarkan dengan hasil sementara yang didapatkan maka ketua kelompok menginstruksikan untuk melakukan ekspreimen model lain untuk menjadi bahan pertimbangan terkait hasil akurasi yang didapatkan. Dengan hasil yang didapatkan kemudian

dilaporkan kepada supervisor terkait hasil pengerjaan yang kemudian dikumpulkan melalui halaman kaggle competition. Kemudian dilanjutkan dengan penyusunan laporan PRO-STEP : Road to Champion yang dimulai dari Bab I hingga BAB IV.

3.3.1 Proses Pelaksanaan

Pada subbab berikut dijelaskan langkah – langkah yang dilakukan selama mengikuti PRO-STEP : Road To Champion dalam mengikuti kompetisi LOGIKA UI 2025. Kompetisi yang berfokus dalam pengembangan model *computer vision* untuk klasifikasi citra budaya di Indonesia menggunakan arsitektur ResNet34. Terdapat beberapa tahapan yang dilakukan dalam proses pengembangan model dimulai dari *data understanding*, *pre – processing*, *modeling* dan evaluasi seperti pada Gambar 3.2 Proses Pengerjaan dan dijelaskan secara detail sebagai berikut :



Gambar 3. 2 Proses Pengerjaan

Berikut Tabel 3.2 Detail Proses Pengerjaan merupakan tabel yang mendefinisi dari proses pelaksanaan berdasarkan Gambar 3.2 Proses Pengerjakan

Tabel 3. 2 Detail Proses Pengerjaan

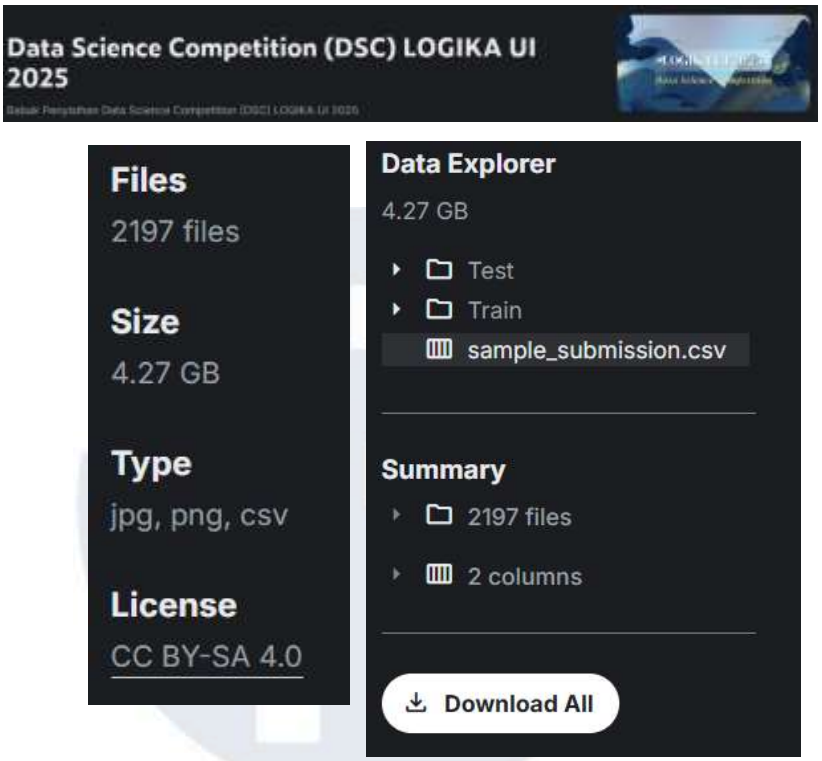
Tahapan	Definisi
Data Understanding	Tahapan awal dengan mengetahui karakteristik dari dataset yang digunakan saat membangun model.

Tahapan	Definisi
	Sepeti mengetahui rasio foto, *channel* warna yang digunakan. Selain itu mengetahui jumlah foto yang terdapat pada *dataset* yang akan digunakan, dan memastikan bahwa file berformat foto yaitu jpg / png.
Pre-Processing	Tahapan kedua saat membangun model yaitu dengan menyamakan atau setiap foto yang digunakan memiliki ukuran dan channel warna yang sama antar foto yang digunakan. Dengan tahapan ini dapat membantu model untuk belajar dengan baik dan membantu untuk model agar beradaptasi dengan model saat belajar.
Modelling	Tahapan ketiga yang menjadi tahapan yang penting dalam membangun model. Tahapan ini untuk menentukan alur model selama model belajar. Pemilihan algoritma yang digunakan pada membangun model terdapat pada tahapan modelling yang akan menentukan hasil akurasi yang didapatkan dengan cara memilih algoritma yang sesuai dengan dataset yang digunakan.
Training	Tahapan keempat yang dilakukan setelah algoritma telah ditentukan dan merupakan proses tahapan model

Tahapan	Definisi
	untuk belajar ataupun beradaptasi dengan dataset untuk mengenali ataupun mengindentifikasi obyek pada dataset.
Evaluation	Tahapan kelima yang dilakukan setelah model telah selesai belajar selama tahapan training. Tahapan evaluasi digunakan untuk mengetahui kinerja model selama melakukan training untuk klasifikasi citra dengan dataset yang diberikan. Untuk mengetahui kinerja model dengan menggunakan classification report yang akan menampilkan metrik seperti akurasi, presisi, recall dan skor F1.

3.3.1.1 Data Understanding

Tahapan ini menjadi tahapan pertama dalam melakukan proses pembangunan model melalui proses pemahaman data. Pada proses *data understanding* yang dilakukan ialah dengan mengunduh dataset yang telah disediakan oleh panitia melalui kaggle.com. Berikut Gambar 3.3 Halaman Kaggle.com merupakan halaman pada kaggle.com untuk mengunduh dataset yang diberikan.



Gambar 3. 3 Halaman Kaggle.com

Berdasarkan Gambar 3.3 Halaman Kaggle.com merupakan tampilan halaman yang terdapat di Kaggle.com yang menunjukkan untuk detail dari *dataset* yang digunakan, jumlah citra yang terdapat pada *dataset*, terdapat ukuran file dari dari *dataset* kemudian terdapat tipe file yang terdapat pada *dataset*, dan selain itu terdapat struktur dari *dataset* yang digunakan seperti *sub-folder* yaitu *test* dan *train*.

```
#jumlah gambar di folder

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. 4 Kode Data Understanding

Berdasarkan Gambar 3.4 Kode Data Understanding menunjukkan penggunaan kode untuk mengetahui jumlah dari masing – masing kategori yang telah ter-label pada folder train dengan memasukkan masing – masing *directory* dari masing – masing kategori seperti (‘./Train/Train/balinese’) merupakan *directory* dari kategori bali, kemudian (‘./Train/Train/batak’) merupakan *directory* dari kategori batak,

kemudian (‘./Train/Train/dayak’) merupakan *directory* dari kategori dayak, kemudian (‘./Train/Train/javanese’) merupakan *directory* dari kategori jawa, kemudian (‘./Train/Train/minangkabau’) merupakan *directory* dari kategori minangkabau. Kemudian terdapat (‘./Test/Test/’) yang merupakan directory dari folder test yang tidak ter-label oleh kategori dari masing – masing asal rumah adat.

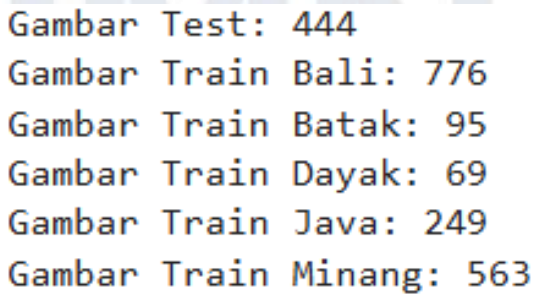
Berdasarkan Gambar 3.4 Kode Data Understanding menunjukkan penggunaan kode untuk mengetahui jumlah citra dari masing – masing *directory* yang telah didefinisikan pada masing- masing variabel. Penggunaan fungsi len() atau length() digunakan untuk menghitung berapa banyak item yang terdapat pada masing – masing variabel yang telah didefinisikan dari masing – masing *directory folder train* maupun *test* seperti len(file_test) yang digunakan untuk mengetahui banyak *file* pada *folder test*, kemudian len(train_bali) yang digunakan untuk mengetahui banyak *file* pada kategori bali, kemudian len(train_batak) yang digunakan untuk mengetahui banyak *file* pada kategori batak, kemudian len(train_dayak) yang digunakan untuk mengetahui banyak *file* pada kategori dayak, kemudian len(train_java) yang digunakan untuk mengatahui banyak *file* pada kategori java, kemudian len(train_minang) yang digunakan untuk mengetahui banyak *file* pada kategori minangkabau.

Berdasarkan Gambar 3.4 Kode Data Understanding menunjukkan penggunaan kode untuk menampilkan output dari masing – masing variabel yang telah didefinisikan seperti print(“Gambar Test:”, jumlah_test) yang digunakan untuk menampilkan *output* dari jumlah *file* pada *folder test*, kemudian print(“Gambar Train Bali:”, jumlah_bali) yang digunakan untuk menampilkan *output* dari jumlah file file dari kategori bali, kemudian print(“Gambar Train Batak:”, jumlah_batak) yang digunakan untuk menampilkan output dari jumlah *file* dari kategori batak, kemudian print(“Gambar Train Dayak”,jumlah_dayak) yang digunakan untuk menampilkan *output* dari jumlah *file* dari kategori batak, kemudian

print(“Gambar Train Java:”, jumlah_java) yang digunakan untuk menampilkan *output* dari jumlah *file* dari kategori java, kemudian print(“Gambar Train Minang:”, jumlah_minang) yang digunakan untuk mengetahui menampilkan *output* dari jumlah *file* dari kategori minang. *Dataset* yang disediakan berukuran 4,23 GB yang berisikan *file* gambar dari *folder train* yang memiliki *sub-folder* yang telah dikategorikan secara spesifik sebagai berikut :

1. Bali memiliki *file* .jpg sebanyak 776.
2. Batak memiliki *file* .jpg sebanyak 95.
3. Dayak memiliki *file* .jpg sebanyak 69.
4. Jawa memiliki *file* .jpg sebanyak 249.
5. Minangkabau memiliki *file* .jpg sebanyak 563.

Maka total 1.752 gambar pada *folder train* dan *folder test* dengan total gambar 444 gambar. Maka total *file* gambar yang disediakan 2.196 gambar, seperti pada Gambar 3.5 Data Understanding yang menunjukkan jumlah dari masing – masing kategori pada dataset.



Gambar 3. 5 Data Understanding

Berikut merupakan definisi dan sampel foto yang terdapat pada dataset dari masing – masing kategori yang terdapat pada dataset sebagai berikut:

1. Gambar 3.6 Sampel Kategori Bali merupakan sampel dataset yang berkategori bali.

Gambar 3. 6 Sampel Kategori Bali

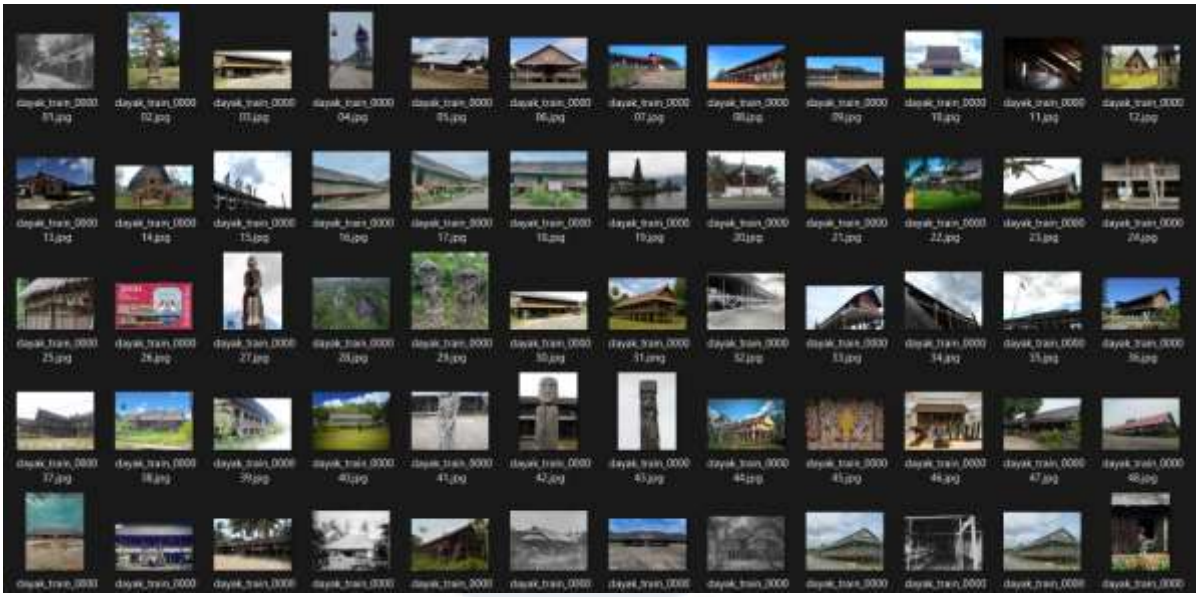
Berdasarkan Gambar 3,6 Sampel Kategori Bali menujukkan bahwa dataset dengan kategori Bali yang terdapat beberapa foto yang berasal dari bali seperti rumah adat, pura, ornamen patung hingga aktifitas masyarakat bali. Dengan didominasi menggunakan rasio potrait dikarenakan obyek foto yang berbentuk menjulang keatas seperti pura ataupun gapura pura dan menggunakan channel warna *black and white.*

2. Gambar 3.7 Sampel Kategori Batak merupakan sampel dataset yang berkategori batak.

Gambar 3. 7 Sampel Kategori Batak

Berdasarkan Gambar 3.7 Sampel Kategori Batak menujukkan bahwa dataset dengan kategori Batak yang terdapat beberapa foto yang berasal dari Batak seperti perangko surat edisi Batak, simbol dari Batak, rumah adat, ornamen tameng dan aktifitas masyakat Batak dengan menggunakan berbagai rasio yaitu landscape dan potrain dan terdapat beberapa foto yang menggunakan channel warna yang berbeda.

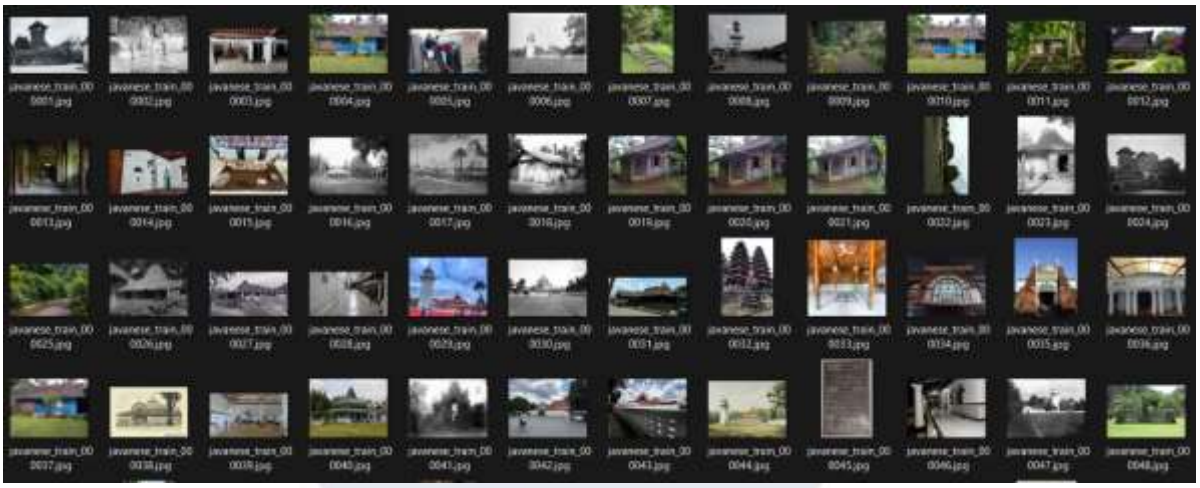
3. Gambar 3.8 Sampel Kategori Dayak merupakan sampel dataset yang berkategori dayak.



Gambar 3. 8 Sampel Kategori Dayak

Berdasarkan Gambar 3.8 Sampel Kategori Dayak menunjukkan dataset dengan kategori Dayak yang terdapat beberapa foto yang berasal dari Dayak seperti rumah adat, perangko surat edisi Dayak, dan patung ciri khas Dayak dengan berbagai rasio dan channel warna yang digunakan.

4. Gambar 3.9 Sampel Kategori Jawa merupakan sampel dataset yang berkategori jawa.



Gambar 3. 9 Sampel Kategori Jawa

Gambar 3.9 Sampel Kategori Jawa menunjukkan dataset dengan kategori Jawa dengan berbagai foto seperti rumah adat, masjid dan surat dengan huruf Jawa, Terdapat beberapa foto yang menggunakan rasio *landscape* dan menggunakan *channel* warna *black and white.*

5. Gambar 3.10 Sampel Kategori Minangkabau merupakan sampel dataset yang berkategori minangkabau.

Gambar 3. 10 Sampel Kategori Minangkabau

Gambar 3.10 Sampel Kategori Minangkabau menujukkan dataset dengan kategori Minangkabau yang terdapat beberapa foto seperti pemandangan, rumah adat, masjid, dan icon di Minangkabau dengan berbagai rasio yang digunakan seperti *landscape* dan *potrat,* selain itu pada channel warna yang digunakan juga terdapat foto dengan *channel* warna *black and white.*

3.3.1.2 Pre-processing

Tahapan *pre-processing* menjadi tahapan yang termasuk dalam rangkaian pengerjaan ataupun tahapan yang penting dalam melakukan proses pembangunan model. Tahapan penting yang harus dilakukan untuk mengolah data mentah menjadi data yang telah diproses agar lebih terstruktur. Tujuan utama dari tahapan *pre-processing* agar *file* memiliki

kesamaan agar saat *training* model, model dapat belajar dengan baik agar dapat memiliki nilai akurasi yang tinggi. Setelah data telah dikumpulkan dan melalui *pre-processing* maka data telah siap digunakan untuk membangun model. Pada tahapan *pre-processing* perlu mencari nilai *mean* dan nilai *standard deviation*, kedua nilai tersebut akan dimanfaatkan untuk tahapan normalisasi data agar setiap gambar memiliki *channel* warna yang sama (R, G, B). Maka dengan itu akan mempermudah model dalam tahapan training karena beban pada model akan ringan dan dapat lebih cepat dan efisien. Pada Gambar 3.11 Kode Pre-Processing merupakan proses untuk mendapatkan *mean* dan *standard deviation* pada dataset di folder training.

```
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
])

dataset = datasets.ImageFolder(root="./Train/Train", transform=transform)
train_loader = DataLoader(dataset, batch_size=16, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False)

mean = 0.
std = 0.
total_img = 0

for images, _ in train_loader:
  batch_samples = images.size(0)
  images = images.view(batch_samples, images.size(1), -1)
  mean += images.mean(2).sum(0)
  std += images.std(2).sum(0)
  total_img += batch_samples

mean /= total_img
std /= total_img

print("Mean:",mean)
print("Std:",std)
```

```
Mean: tensor([0.4680, 0.4690, 0.4481])
Std: tensor([0.2350, 0.2394, 0.2528])
```

Gambar 3. 11 Kode Pre-Processing

Berdasarkan Gambar 3.11 Kode Pre-Processing menunjukkan bahwa pada variabel *transform* terdapat fungsi *resize* untuk ukuran gambar menjadi 224x224 piksel dikarenakan pada arsitektur yang dipilih mengharuskan untuk setiap *output* agar memiliki ukuran 224 piksel kemudian terdapat fungsi ToTensor() yang digunakan untuk mengubah format standar yaitu .jpg menjadi berformat **tensor**. Kemudian terdapat variabel dataset = datasets.ImageFolder(root="./Train/Train", transform = transform) yang digunakan untuk memuat file yang terdapat di *folder train*, kemudian terdapat variabel train_loader = DataLoader(dataset, batch_size = 16, shuffle = true) yang digunakan untuk memuat *file* pada *folder train* kemudian dibagi menjadi 16 *batch* agar menjadi ringan jika saat digunakan kemudian file akan diacak supaya gambar menjadi tidak sesuai urutan saat model dilatih secara terus menerus. Kemudian gambar akan diproses untuk dapat mencari nilai *mean* atau rata – rata dan nilai *standard deviation* atau penyebaran, kode ini akan diulang – ulang hingga 16 *batch* agar setiap gambar selesai diproses, setelah mendapatkan nilai mean pada setiap gambar kemudian akan di jumlah yang akan disimpan pada variabel total_img.

Dengan proses yang dilakukan diatas dengan menghitung *mean* dan *standar deviation* dari setiap *channel* warna pada gambar di *folder Train* mendapatkan hasil bahwa *mean* : ([0,4682, 0.4690, 0.4481]) dan *standar deviation* : ([0.2350, 0.2394, 0.2528]). Maka dengan hasil yang didapatkan digunakan untuk menormalisasi citra pada *folder test* maupun *train*.

Setelah menghitung *mean* dan *standard deviation* untuk yang digunakan menormalisasi citra kemudian melakukan proses tranformasi pada *folder test* maupun *train* seperti pada Gambar 3.12 Kode Transformasi Data

```
train_transform = transforms.Compose([
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(), #ubah menjadi HWC -> CHW
    transforms.Normalize(
        mean=[0.4680, 0.4690, 0.4481],
        std=[0.2350, 0.2394, 0.2528]
    )
])

test_transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.4680, 0.4690, 0.4481],
        std=[0.2350, 0.2394, 0.2528]
    )
])
```

Gambar 3. 12 Kode Transformasi Data

Proses ini dilakukan agar setiap citra yang digunakan untuk melatih model agar memiliki kriteria yang sama, dengan proses ini akan mempermudah dalam melatih model. Dalam tahapan *pre-processing*, terdapat beberapa transformasi pada setiap citra agar meningkatkan nilai kualitas data dan menyesuaikan agar data memiliki format yang sama seperti sebagai berikut: Proses tersebut seperti

1. RandomResizedCrop merupakan library yang terdapat pada pytorch yang digunakan pada *deep learning*. Teknik digunakan untuk memotong gambar input dengan resolusi 224 x 224 piksel dari bagian gambar tersebut secara acak. Teknik ini digunakan agar citra memiliki format yang sama dari segi resolusi.

2. RandomHorizontalFlip merupakan teknik yang digunakan dalam membangun model pada *deep learning* digunakan untuk membalikkan citra agar menjadi horizontal seperti efek pada

cermin.

3. ToTensor yang digunakan untuk mengubah citra menjadi tensor ataupun mengubah warna atau channel pada citra yang awalnya *Height, Width, Channel* (HWC) menjadi *Channel, Width, Height* (CHW) selain itu juga digunakan untuk mengubah rentang nilai piksel dari 0-255 (integer) menjadi 0.0-1.0 (float),

4. Normalize yang digunakan untuk menormalisasikan pada *file train* dengan menggunakan nilai *mean* dan nilai *standar deviation* yang sudah didapatkan pada tahapan sebelumnya.

Tahapan transformasi data tidak hanya terdapat pada file train saja namun pada file test juga diperlukan transformasi data agar memiliki format yang sama, terdapat beberapa tahapan test_transform seperti:

1. Resize merupakan teknik digunakan untuk memotong gambar input dengan resolusi 224 x 224 piksel dari bagian gambar tersebut.

2. CenterCrop merupakan teknik yang digunakan saat membangun model *deep learning*, dimana teknik ini akan memotong tengah gambar input dengan rasio 224 x 224 piksel agar hasil pengujian tetap konsisten agar dapat mencapai hasil testing dengan akurasi yang baik, terdapat

3. Normalize yang digunakan untuk menormalisasikan pada *file test* dengan menggunakan nilai *mean* dan nilai *standard deviation* yang sudah didapatkan pada tahapan sebelumnya agar dapat mencapai nilai konvergensi lebih cepat pada pelatihan model.

Setelah beberapa tahapan yang telah dilakukan pada transformasi data, data telah memiliki format yang sama pada *folder train* maupun *folder test.* Maka data telah dapat digunakan untuk membangun model untuk membangun model klasifikasi citra dengan memiliki akurasi yang tinggi

dan model akan belajar dengan baik karena menggunakan dataset yang memiliki format yang terstruktur.

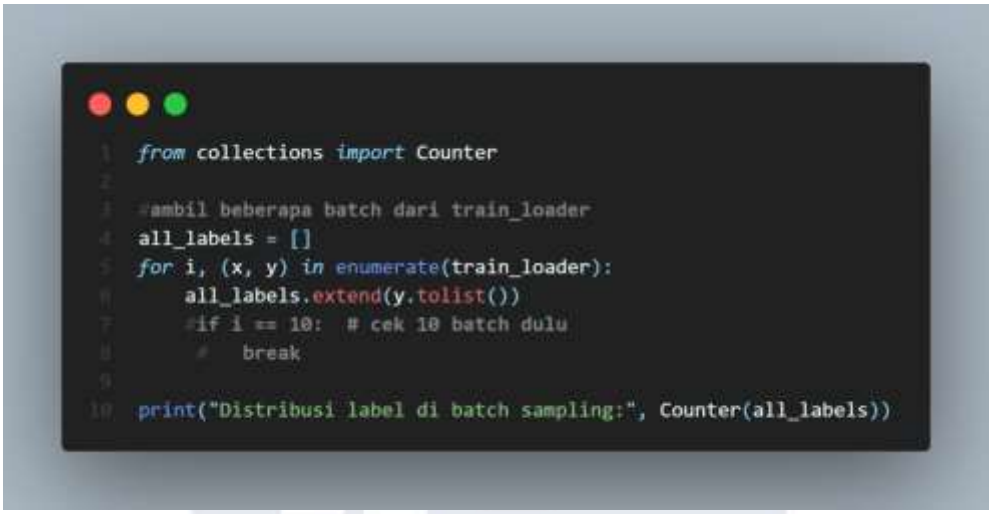
Setelah tahapan *collecting data* dan *transform* data tahapan berikutnya ialah proses *balancing* menggunakan WeightedRandomSampler, dikarenakan saat distribusi data pada *dataset* yaitu ketidakseimbangan penyebaran data pada setiap kategori yang dapat disebut dengan *imbalanced dataset*, dimana selisih jumlah pada distribusi data yang sangat signifikan yang berpotensi terjadinya model menjadi bias dan model cenderung hanya akan memprediksi jumlah kategori yang mayoritas saja, akan mempengaruhi kinerja model saat melakukan training seperti pada Gambar 3.13 Kode Balancing Data agar setiap kelas memiliki probabilitas bobot yang sama saat melakukan training supaya model dapat berlatih secara adil agar tidak hanya terfokus pada salah satu kelas yang memiliki bobot yang paling banyak.

```python
from torch.utils.data import WeightedRandomSampler

# agar data dari masing - masing kelas tidak memiliki selisih yang terlalu jauh
# balancing
# hitung sample weight untuk setiap data di train_dataset
class_counts = [776, 95, 69, 249, 563]  # dari counter kamu
weights = 1. / torch.tensor(class_counts, dtype=torch.float)
samples = train_dataset.samples
sample_weights = [weights[label] for _, label in samples]

sampler = WeightedRandomSampler(sample_weights, num_samples=len(sample_weights), replacement=True)

train_loader = DataLoader(train_dataset, batch_size=32, sampler=sampler)
```



```
from collections import Counter

ambil beberapa batch dari train_loader
all_labels = []
for i, (x, y) in enumerate(train_loader):
    all_labels.extend(y.tolist())
    if i == 10:  # cek 10 batch dulu
        break

print("Distribusi label di batch sampling:", Counter(all_labels))
```

```
Distribusi label di batch sampling: Counter({4: 380, 0: 351, 3: 350, 2: 344, 1: 327})
```

Gambar 3. 13 Kode Balancing Data

Berdasarkan Gambar 3.13 Kode Balancing Data menujukkan bahwa pada variabel class_counts yang menunjukkan jumlah gambar pada setiap kategori seperti :

1. kelas 0 (Bali) memiliki 776 gambar yang dapat dianggap sebagai kategori mayoritas karena memiliki jumlah yang paling banyak dibandingkan kategori lainnya.
2. kelas 1 (Batak) memiliki 95 gambar.
3. kelas 2 (Dayak) memiliki 69 gambar.
4. kelas 3 (Jawa) memiliki 249 gambar.
5. kelas 4 (Minangkabau) memiliki 563 gambar.

Penggunaan train_loader = DataLoader(train_dataset, batch_size = 32, sampler = sampler) variabel train_loader terdapat fungsi yang dapat melakukan *WeightedRandomSampler* pada *folder train* yang dibagi menjadi 32 *batch*. Setelah proses balancing berhasil maka bobot setiap kelas yang awal mula nya tidak seimbang menjadi seimbang atau adil seperti sebagai berikut :

1. Kelas 0 (Bali) berjumlah 351 gambar.
2. Kelas 1 (Batak) berjumlah 327 gambar.
3. Kelas 2 (Dayak) berjumlah 344 gambar.

4. Kelas 3 (Jawa) berjumlah 350 gambar.
5. Kelas 4 (Minangkabau) berjumlah 380 gambar.

Melalui hasil distribusi yang didapatkan maka data sudah cukup seimbang atau merata jika akan digunakan untuk melatih model, dibandingkan sebelum melakukan *balancing* data yang dapat dikatakan bahwa data tidak seimbang karena memiliki selisih yang cukup jauh pada setiap kelasnya untuk digunakan melatih model klasifikasi citra.

3.3.1.3 Modeling

Tahapan ini dilakukan untuk membangun model Resnet34. Berikut pada Gambar 3.14 Kode Model ResNet34 merupakan kode yang digunakan dalam pengembangan pada base model dengan menggunakan Resnet34.

```python
import torchvision
import torch.nn as nn
import torch.optim as optim

class_weights = 1. / torch.tensor(class_counts, dtype=torch.float)
model = torchvision.models.resnet34(pretrained=True)

# freeze backbone
for param in model.parameters():
    param.requires_grad = False

# ganti classifier sesuai jumlah kelas
num_classes = len(train_dataset.classes)
model.fc = nn.Linear(model.fc.in_features, num_classes)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

criterion = nn.CrossEntropyLoss(weight=class_weights.to(device))
optimizer = optim.AdamW(model.fc.parameters(), lr=1e-3, weight_decay=1e-2)
```

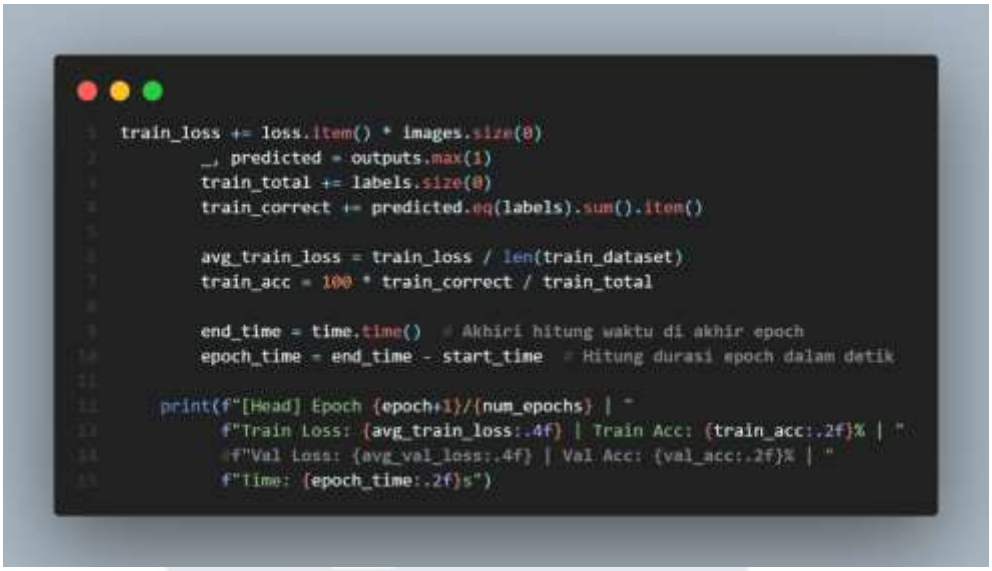
Gambar 3. 14 Kode Model ResNet34

Berdasarkan Gambar 3.14 Kode Model ResNet34 yang menunjukan kode yang digunakan untuk pengembangan model dengan menggunakan transfer learning dengan menggunakan Resnet34. Dengan

penggunaan *transfer learning* memudahkan dalam melatih model menggunakan fitur – fitur yang telah dipelajari dari dataset dengan skala yang besar, penggunaan ini akan lebih efisien dibandingkan melatih model dari awal yang akan membutuhkan banyak waktu. Dimulai dengan melakukan *freeze* pada *backbone* model dengan menggunakan param.requires_grad = false yang digunakan agar dapat menggunakan fitur yang telah terdapat pada model. Kemudian dilakukan modifikasi pada layer terakhir atau *Fully Connected* agar jumlah output agar sesuai dengan jumlah kategori pada *dataset*. Penggunakan *CrossEntropyLoss* pada *criterion* merupakan tahapan selanjutnya dari proses *balancing* data yang digunakan mengukur hasil prediksi model pada label. Dengan menghitung *loss* yang didapatkan akan mudah untuk mengetahui seberapa mampu untuk model dalam melakukan prediksi, jika memiliki nilai *loss* yang rendah maka dapat dikatakan bahwa model dapat mengklasifikasikan citra dengan baik, bukan bias pada kategori yang mayoritas. Selain itu terdapat **AdamW** sebagai *optimizer* yang berfungsi meminimalisir adanya *overfit* dengan menggunakan *weightdecay* pada model dan memastikan bahwa model dapat berlatih dengan stabil dan generalisasi dengan baik.

3.3.1.4 Training

Tahapan ini dilakukan untuk melatih model dengan data train yang terdapat pada *dataset*. Selain itu digunakan untuk mengetahui proses dari setiap *epoch* saat melakukan *training* yang dilihat *Traning Loss* dan *Training Accuracy*. Berikut pada Gambar 3.15 Training ResNet34 merupakan kode dari training pada ResNet34.

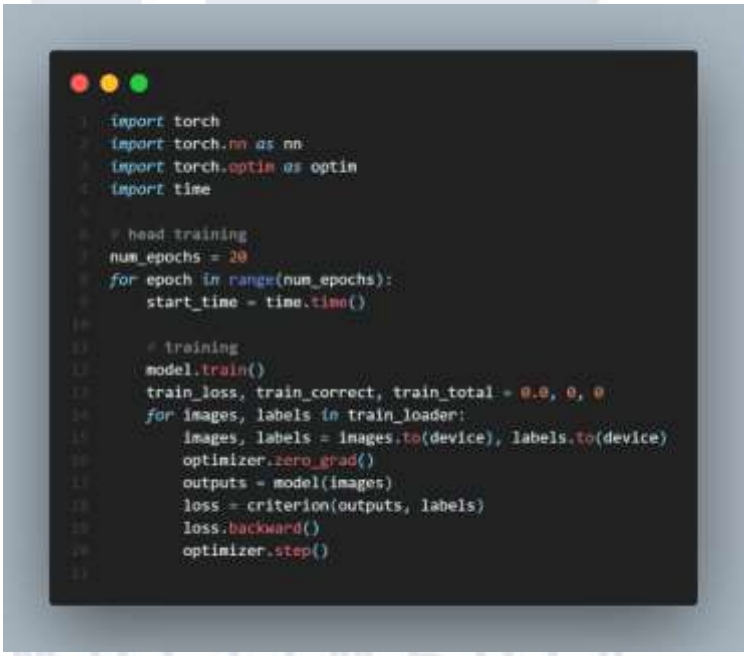


```
        train_loss += loss.item() * images.size(0)
        _, predicted = outputs.max(1)
        train_total += labels.size(0)
        train_correct += predicted.eq(labels).sum().item()

        avg_train_loss = train_loss / len(train_dataset)
        train_acc = 100 * train_correct / train_total

        end_time = time.time()  # Akhiri hitung waktu di akhir epoch
        epoch_time = end_time - start_time  # Hitung durasi epoch dalam detik

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



```
import torch
import torch.nn as nn
import torch.optim as optim
import time

# head training
num_epochs = 20
for epoch in range(num_epochs):
    start_time = time.time()

    # training
    model.train()
    train_loss, train_correct, train_total = 0.0, 0, 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

Gambar 3. 15 Training ResNet34

Gambar 3.15 Training ResNet34 merupakan tampilan dari training dengan 20 epoch. Setiap epoch akan diproses pada model.train() untuk mengaktifkan lapisan dropout, kemudian akan diproses menggunakan GPU agar mempercepat komputasi. Penggunaan optimizer.step() digunakan untuk model dapat mengenali rumah adat secara lebih akurat. Penggunaan train_acc = 100 * train_correct /

train_total digunakan untuk dapat menghitung rata – rata loss dan akurasi pada setiap epoch agar dapat mengetahui stabilitas atau presentase pada performa model saat melakukan training secara realtime. Penggunaan epoch_time = end_time – start time, digunakan untuk mengetahui durasi setiap epoch saat melakukan training, dengan mengetahui durasi pada setiap epoch dapat digunakan untuk mengetahui performa yang dapat dilakukan oleh GPU untuk melakukan komputasi dari setiap epoch. Training dilakukan untuk menerapkan transfer learning pada ResNet34 dengan Pelatihan ini dilakukan bertujuan agar model belajar dapat mengklasifikasikan obyek yang disediakan yaitu budaya Indonesia. Penerapan ini sangat membantu dalam melakukan efisiensi secara durasi dan model dapat akan belajar secara stabil dibandingkan menggunakan model dari scratch. Berikut Gambar 3.16 Hasil Training ResNet34 merupakan hasil yang didapatkan saat training pada head dengan 20 epoch

```
[Head] Epoch 1/20 | Train Loss: 1.1030 | Train Acc: 29.74% | Time: 359.91s
[Head] Epoch 2/20 | Train Loss: 0.8049 | Train Acc: 44.46% | Time: 357.62s
[Head] Epoch 3/20 | Train Loss: 0.6852 | Train Acc: 52.74% | Time: 353.96s
[Head] Epoch 4/20 | Train Loss: 0.6074 | Train Acc: 58.73% | Time: 369.58s
[Head] Epoch 5/20 | Train Loss: 0.5955 | Train Acc: 59.47% | Time: 342.60s
[Head] Epoch 6/20 | Train Loss: 0.5178 | Train Acc: 64.50% | Time: 340.37s
[Head] Epoch 7/20 | Train Loss: 0.5155 | Train Acc: 65.58% | Time: 354.03s
[Head] Epoch 8/20 | Train Loss: 0.4533 | Train Acc: 69.58% | Time: 345.19s
[Head] Epoch 9/20 | Train Loss: 0.4361 | Train Acc: 69.12% | Time: 345.01s
[Head] Epoch 10/20 | Train Loss: 0.3885 | Train Acc: 72.03% | Time: 348.13s
[Head] Epoch 11/20 | Train Loss: 0.3784 | Train Acc: 73.17% | Time: 343.60s
[Head] Epoch 12/20 | Train Loss: 0.3955 | Train Acc: 70.55% | Time: 349.02s
[Head] Epoch 13/20 | Train Loss: 0.3917 | Train Acc: 71.18% | Time: 350.99s
[Head] Epoch 14/20 | Train Loss: 0.3509 | Train Acc: 72.89% | Time: 340.32s
[Head] Epoch 15/20 | Train Loss: 0.3381 | Train Acc: 75.29% | Time: 351.38s
[Head] Epoch 16/20 | Train Loss: 0.3711 | Train Acc: 74.43% | Time: 368.27s
[Head] Epoch 17/20 | Train Loss: 0.3351 | Train Acc: 74.66% | Time: 347.83s
[Head] Epoch 18/20 | Train Loss: 0.3262 | Train Acc: 75.91% | Time: 347.18s
[Head] Epoch 19/20 | Train Loss: 0.3262 | Train Acc: 75.23% | Time: 353.81s
[Head] Epoch 20/20 | Train Loss: 0.3109 | Train Acc: 76.03% | Time: 361.82s
```

Gambar 3. 16 Hasil Training ResNet34

Berdasarkan Gambar 3.16 Hasil Training ResNet34 hasil akhir dengan 20 *epoch* yang telah didapatkan menghabiskan waktu sekitar 2 jam

dengan train accuracy 76.03% dengan training loss 0.3109. Akurasi yang didapatkan pada *epoch* ke – 3 hingga ke – 20 mempunyai selisih yang tidak jauh setiap epoch, meskipun pada *epoch* ke – 1 hingga *epoch* ke – 2 memiliki selisih sekitar 15% yang merupakan akurasi yang wajar saat permulaan model belajar. Pada *epoch* 10 sampai 20 mendapatkan akurasi yang tidak terlalu jauh, selain itu pada setiap *epoch* memiliki durasi dengan rata – rata 350 detik yang dapat ditunjukkan bahwa model belajar dengan stabil, tanpa adanya selisih durasi yang terlalu jauh. Dengan hasil akhir train accuracy 76.03%, maka diperlukan *fine tuning* untuk dapat meningkatkan presentase akurasi pada model saat belajar.

Gambar 3. 17 Kode Optimasi

Berdasarkan Gambar 3.17 Kode Optimasi menunjukkan bahwa layer terakhir yaitu layer4 dan fc yang akan dilatih tidak pada keseluruhan model, dilatih agar model dapat mengklasifikan obyek dengan baik, maka akan menghasilkan nilai akurasi yang cukup tinggi. Terdapat perbedaan dari penggunaan *optimizer* yang digunakan untuk mengatur learning rate dari layer4 dan layer FC, seperti pada model.layer4.parameters() dengan menggunakan learning rate = 1e-4 (0.0001) dan model.fc.parameters() dengan menggunakan learning rate = 1e-3 (0.001), hal ini digunakan untuk mencegah perubahan pada model secara signifikan dan untuk mengurangi terjadi risiko *overfit* yang dapat terjadi. Perubahan learning rate diperlukan dikarenakan pada layer4 memiliki fitur *high-level* yang terdapat pada *pretrained* yang telah dilatih pada **ImageNet** yang memiliki bobot yang kuat, yang dapat menyebabkan model menjadi tidak seimbang selama training yang menyebabkan model kurang belajar terhadap dataset yang digunakan, maka dengan itu learning rate pada layer FC diatur lebih besar agar model dapat lebih belajar atau menyesuaikan dengan dataset yang digunakan. Learning rate menjadi aspek penting untuk mengatur setiap layer agar model dapat belajar dengan baik. Penggunaan param.requires_grad = True difungsikan agar layer tersebut dapat belajar. Terdapat beberapa yang optimasi yang dilakukan berdasarkan Gambar 3.17 Kode Optimasi berikut merupakan Tabel 3.3 Optimasi penjelasan secara detail dari optimasi yang dilakukan pada kode optimasi untuk dapat meningkatkan nilai akurasi pada model.

Tabel 3. 3 Optimasi

Tahapan	Penjelasan
Unfreezing layer 4 dan layer fc	Membuka / mengaktifkan layer4 dan layer fc sehingga bobot pada layer tersebut diperbarui selama training dilakukan.

Tahapan	Penjelasan
Different Learning Rate	Terdapat perbedaan *learning rate* pada layer4 dan layer fc, untuk layer4 memiliki *learning rate* yang relatif kecil yaitu 1e-4 yang digunakan untuk menyesuaikan dengan data yang baru. Pada *layer* fc memiliki *learning rate* yang relatif lebih besar yaitu 1e-3 yang digunakan agar pada layer ini cepat beradaptasi atau menyesuaikan dengan *dataset* yang digunakan.

Dengan melakukan teknik *fine tuning* diharapkan bahwa model dapat belajar tanpa merusak pada layer awal. Berikut hasil yang didapatkan saat *fine tuning* dengan 20 *epoch*.

```
[FineTune] Epoch 1/20 | Train Loss: 0.3140 | Train Acc: 78.54% | Time: 352.39s
[FineTune] Epoch 2/20 | Train Loss: 0.1382 | Train Acc: 90.81% | Time: 358.63s
[FineTune] Epoch 3/20 | Train Loss: 0.0849 | Train Acc: 93.95% | Time: 344.19s
[FineTune] Epoch 4/20 | Train Loss: 0.0507 | Train Acc: 96.80% | Time: 354.57s
[FineTune] Epoch 5/20 | Train Loss: 0.0484 | Train Acc: 97.32% | Time: 353.22s
[FineTune] Epoch 6/20 | Train Loss: 0.0535 | Train Acc: 96.75% | Time: 396.64s
[FineTune] Epoch 7/20 | Train Loss: 0.0359 | Train Acc: 98.00% | Time: 497.27s
[FineTune] Epoch 8/20 | Train Loss: 0.0336 | Train Acc: 98.23% | Time: 519.62s
[FineTune] Epoch 9/20 | Train Loss: 0.0293 | Train Acc: 98.40% | Time: 481.03s
[FineTune] Epoch 10/20 | Train Loss: 0.0368 | Train Acc: 97.49% | Time: 489.17s
[FineTune] Epoch 11/20 | Train Loss: 0.0512 | Train Acc: 97.60% | Time: 497.61s
[FineTune] Epoch 12/20 | Train Loss: 0.0243 | Train Acc: 98.40% | Time: 522.31s
[FineTune] Epoch 13/20 | Train Loss: 0.0222 | Train Acc: 98.63% | Time: 528.16s
[FineTune] Epoch 14/20 | Train Loss: 0.0286 | Train Acc: 98.46% | Time: 511.26s
[FineTune] Epoch 15/20 | Train Loss: 0.0219 | Train Acc: 98.74% | Time: 477.08s
[FineTune] Epoch 16/20 | Train Loss: 0.0211 | Train Acc: 98.97% | Time: 460.46s
[FineTune] Epoch 17/20 | Train Loss: 0.0245 | Train Acc: 98.57% | Time: 495.67s
[FineTune] Epoch 18/20 | Train Loss: 0.0202 | Train Acc: 98.92% | Time: 489.40s
[FineTune] Epoch 19/20 | Train Loss: 0.0207 | Train Acc: 98.34% | Time: 484.68s
[FineTune] Epoch 20/20 | Train Loss: 0.0243 | Train Acc: 98.12% | Time: 466.93s
```

Gambar 3. 18 Hasil Training Fine-Tuning

Berdasarkan Gambar 3.18 Hasil Training Fine-Tuning

menunjukkan bahwa model melanjutkan akurasi saat *epoch* terakhir pada training pada c*lassification head* sebelumnya di train accuracy 76.03%. Hasil akhir dengan 20 *epoch* yang didapatkan menghabiskan waktu 2 jam 30 menit dengan *train accuracy* 98.12% dengan *training loss* 0.0243. Hasil tersebut memiliki selisih 22.09% dari *training* sebelumnya. Dengan selisih *epoch* yang tidak terlalu banyak, maka model telah belajar dengan baik dan telah menyesuaikan dengan dataset yang digunakan. Penerapan *fine tuning* pada *layer*-4 memiliki dapat yang efektif dalam melakukan adaptasi pada obyek . Meskipun telah mendapatkan hasil akhir *train accuracy* 98.12%.

Terdapat perbedaan pada kode training yang telah dilakukan yaitu dari segi optimasi yang menyebabkan nilai akurasi yang berbeda yang didapatkan dari kedua training yang telah dilakukan. Dengan melakukan beberapa perubahan pada learning rate yang terdapatkan pada layer yang diterapkan pada tahapan training, memiliki presentase akurasi yang didapatkan seperti pada Tabel 3.4 Perbandingan Model.

.

Tabel 3. 4 Perbandingan Model

Fitur	Sebelum Fine-Tuning	Setelah Fine-Tuning
Struktur	Single Learning Rate yang merupakan layer pada model memiliki nilai learning rate yang sama.	Different Learning Rate yang merupakan layer pada model memiliki nilai learning rate yang berbeda antar layer yang memiliki learning rate yang berbeda.
Optimizer	AdamW	AdamW

Fitur	Sebelum Fine-Tuning	Setelah Fine-Tuning
Learning Rate	Parameter yang digunakan pada layer fully connecter 1e-3	Penggunaan parameter yang berbeda seperti pada layer4 yang menggunakan 1e-4 dan pada fully connecter yang menggunakan 1e-3.
Weight Decay	Penggunaan weight decay dengan nilai 1e-2.	Penggunaan weight decay dengan nilai default
Presentase Akurasi	Presentase akurasi yang dihasilkan selama 20 epoch adalah 76.03%	Presentase akurasi yang dihasilkan selama 20 epoch adalah 98.12%.

Berdasarkan Tabel 3.4 Perbandingan Model menunjukkan bahwa optimasi yang telah dilakukan dapat meningkatkan nilai akurasi dengan nilai epoch pada training yang sam pada model yang dibangun memiliki selisih sekitar 20% pada setelah optimasi dan sebelum dioptimasi.

Berikut merupakan Tabel 3.5 Perbandingan Epoch pada Model yang akan menunjukkan secara detail terkait perbandingan nilai akurasi dan loss dengan menggunakan model ResNet34 saat sebelum dilakukan fine-tuning dan setelah fine-tuning.

Tabel 3. 5 Perbandingan Epoch pada Model

Epoch	Sebelum Fine-Tuning		Setelah Fine-Tuning	
	Loss	Accuracy	Loss	Accuracy
Epoch – 1	1.1030	29.74%	0.3140	78.54%
Epoch – 5	0.5955	59.47%	0.0484	97.32%
Epoch – 10	0.3885	72.03%	0.0368	97.49%
Epoch - 20	0.3109	76.03%	0.0243	98.12%

Berdasarkan Tabel 3.5 Perbandingan Epoch pada Model dapat menunjukkan bahwa penggunaan metode fine-tuning dapat membantu adanya peningkatan akurasi pada model, dengan mendapatkan selisih 22.03% pada epoch – 20. Dengan nilai perbandingan yang seimbang dari epoch yang dijalankan memiliki epoch yang sama pada kedua pembanding.

3.3.1.5 Evaluation

Tahapan berikut merupakan menjadi tahapan terakhir untuk mengetahui kinerja model dalam melakukan klasifikasi citra budaya di Indonesia. Dengan menggunakan *classification report* memudahkan untuk melihat kinerja model berdasarkan kelas pada data *train*. Berikut Gambar 3.19 Evaluasi merupakan *classification report* pada sebagai nilai evaluasi dari kinerja model untuk melakukan klasifikasi citra budaya di Indonesia.

Classification Report Base Model:

	precision	recall	f1-score	support
balinese	1.00	0.97	0.98	365
batak	0.97	1.00	0.99	348
dayak	0.99	1.00	0.99	364
javanese	0.97	0.98	0.98	312
minangkabau	0.99	0.97	0.98	363
accuracy			0.98	1752
macro avg	0.98	0.98	0.98	1752
weighted avg	0.98	0.98	0.98	1752

Gambar 3. 19 Evaluasi

Berdasarkan Gambar 3.19 Evaluasi menunjukkan bahwa *base* model telah belajar dengan baik dengan rata – rata *f1- score* mencapai 98 %, dengan nilai akurasi 98%. Berikut merupakan rincian dari kemampuan model dalam melakukan klasifikasi gambar pada setiap kategori :

1. Bali merupakan kelas yang memiliki nilai presisi tertinggi dengan nilai presisi 1.00, nilai *recall* 0.97, *f1-score* 0,98 dan dapat mengklasifikasikan citra 365 citra.
2. Batak menjadi kelas yang memiliki nilai presisi 0.97, nilai *recall* 1.00, *f1-score* 0.99 dan dapat mengklasifikasikan 348 citra.
3. Dayak menjadi kelas dengan nilai presisi 0.99, nilai *recall* 1.00, *f1-score* 0.99 dan dapat mengklasifikasikan 364 citra.
4. Jawa yang memiliki nilai presisi 0.97, nilai *recall* 0.98, *f1-score* 0.98 dan dapat mengklasifikasikan 312 citra.
5. Minangkabau yang memiliki nilai presisi 0.99, nilai *recall* 0.97, *f1-score* 0.97 dan dapat mengklasifikasikan 363 citra.

Secara keseluruhan hasil evaluasi yang didapatkan maka model telah mampu mengklasifikasikan 1752 gambar dari 5 kategori.

Dengan keseluruhan nilai evaluasi yang didapatkan maka model telah belajar dengan baik karena telah dapat mengklasifikan citra dari macam – macam rumah adat yang dapat dilihat dari kolom support dan model tidak

terlalu menonjol pada hanya kelas yang memiliki data paling banyak.

3.3.2 Kendala yang Ditemukan

Berikut merupakan kendala yang ditemukan selama proses pengerjaan dalam PRO-STEP : Road to Champion Program :

1. Ketidaksamaan fokus kategori kompetisi yang diikuti (data science) dengan fokus peminatan yang diambil (database).
2. Hambatan pada tahap pemodelan dikarenakan karakteristik dataset yang digunakan, maka sulitnya menemukan model yang relevan dengan dataset yang tidak seimbang maka harus diperlukan metode lain untuk dapat membantu agar model dapat relevan.
3. Ketergantungan dengan penggunaan Google Colab yang sangat membantu dalam pengerjaan dengan cara kolaborasi antar anggota namun dalam penggunaan nya sangat bergantung dengan stabilitas sinyal internet yang mengurangi waktu pengerjaan.
4. Kurangnya kejelasan informasi yang didapatkan pada saat mengikuti *Technical Meeting* mengenai prosedur atau mekanisme pada tahapan pengumpulan file pada sistem penilaian pada platform **Kaggle Competition**.

3.3.3 Solusi atas Kendala yang Ditemukan

Berikut merupakan solusi atas kendala yang ditemukan selama proses pengerjaan dalam PRO-STEP: Road to Champion:

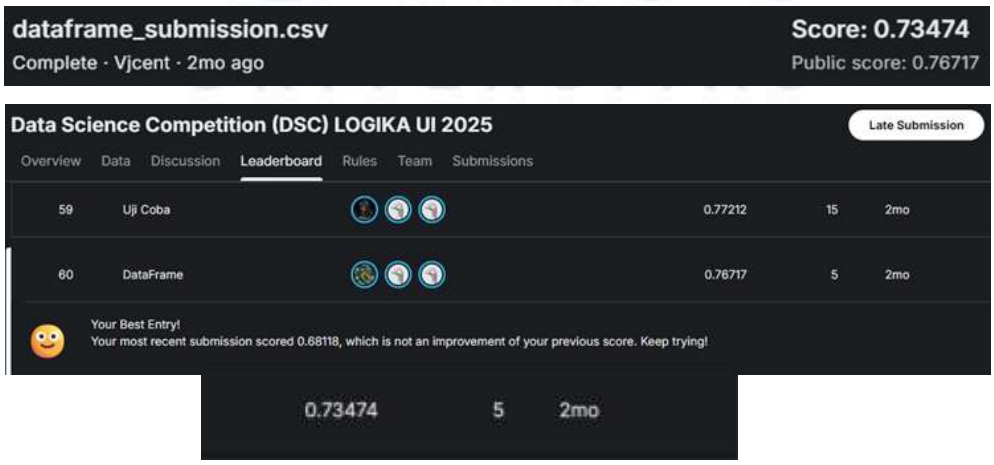
1. Melakukan *review* atau memahami ulang modul terkait deep learning dan memahami ulang terkait projek klasifikasi citra yang telah dikerjakan dan melihat video terkait membangun model klasifikasi citra di Internet.
2. Memahami karakteristik pola *dataset* secara mandiri dan melakukan beberapa eksperimen yang dilakukan untuk mengetahui beberapa model yang cocok dan memiliki akurasi yang tinggi, selan itu

melakukan metode *balancing data* agar model dapat belajar dengan baik dan secara adil tidak cenderung pada salah satu kategori saja.

3. Menerapkan tahapan pengerjaan secara offline dengan penggunaan **Jupyter Notebook** untuk penulisan kode pada saat tahapan awal hingga akhir selama membangun model.
4. Melakukan percobaan secara mandiri untuk tahapan pengumpulan hasil pekerjaan dengan memahami dokumentasi dari **Kaggle Competition** dan melakukan beberapa penggunggahan untuk mengetahui format file yang digunakan.

3.4 Hasil Lomba/Kompetisi

Setelah melakukan beberapa tahapan dari *collecting data, pre-processing data, modeling data, traning data* hingga *evaluation data*. Maka dengan itu tahapan proses pembuatan model untuk klasifikasi citra rumah adat di Indonesia telah selesai dan mendapatkan nilai akurasi yang tinggi namun masih mendapatkan nilai yang kurang saat melakukan pengumpulan pada halaman LOGIKA UI 2025 di kaggle.com. Berikut Gambar 3.20 Hasil Kompetisi merupakan hasil leaderboard ataupun peringkat yang didapatkan setelah melakukan pengumpulan pada halaman Data Science Competition LOGIKA UI 2025 di Kaggle.com



Gambar 3. 20 Hasil Kompetisi

Gambar 3.20 Hasil Kompetisi menunjukkan hasil *leaderboard* saat hasil pengerjaan telah dikumpulkan melalui Kaggle.com, dengan mendapat urutan 60 dari total partisipan 94 kelompok. Dengan mendapatkan hasil *score* 0.73474 dengan dan *public score* 0.76717. Kemudian terdapat file dengan format csv yang disusun yaitu 2 kolom yaitu: id yang merupakan nama file dan style merupakan kategori dari klasifikasi citra budaya Indonesia seperti Gambar 3.21 Submit CSV

id	style
Test_001	balinese
Test_002	minangkabau
Test_003	javanese
Test_004	balinese
Test_005	balinese
Test_006	balinese
Test_007	balinese
Test_008	balinese
Test_009	javanese
Test_010	minangkabau

Gambar 3. 21 Submit CSV

Gambar 3.21 Submit CSV dapat menunjukkan bahwa hasil prediksi pada Test_001 hingga Test_010 terhadap model yang telah dilakukan seperti Test_001 merupakan kategori bali, Test_002 merupakan kategori minangkabau, Test_003 merupakan kategori jawa, Test_003 merupakan kategori jawa, Test_004 merupakan kategori bali, Test_005 merupakan kategori bali, Test_006 merupakan kategori bali, Test_007 merupakan kategori bali, Test_008 merupakan kategori bali, Test_009 merupakan kategori jawa, dan Test_010 merupakan kategori minangkabau.

Terdapat sertifikat yang diberikan oleh penyelenggara atas apresiasi telah mengikuti Data Science Competition (DSC) LOGIKA UI 2025 seperti pada Gambar 3.22 Sertifikat Kompetisi.

Gambar 3. 22 Sertifikat Kompetisi

Sertifikat ini diperoleh setelah menyelesaikan kompetisi dari awal hingga akhir selama kompetisi berlangsung. Menjadi bukti bahwa peserta telah menyelasaikan kompetisi dengan baik dan secara supportif tidak melanggar prosedur dari penyelenggara kompetisi.