PALM **RECOGNITION**

Financial Technology

Partisipan Kelompok

Defrizal Yahdiyan Risyad

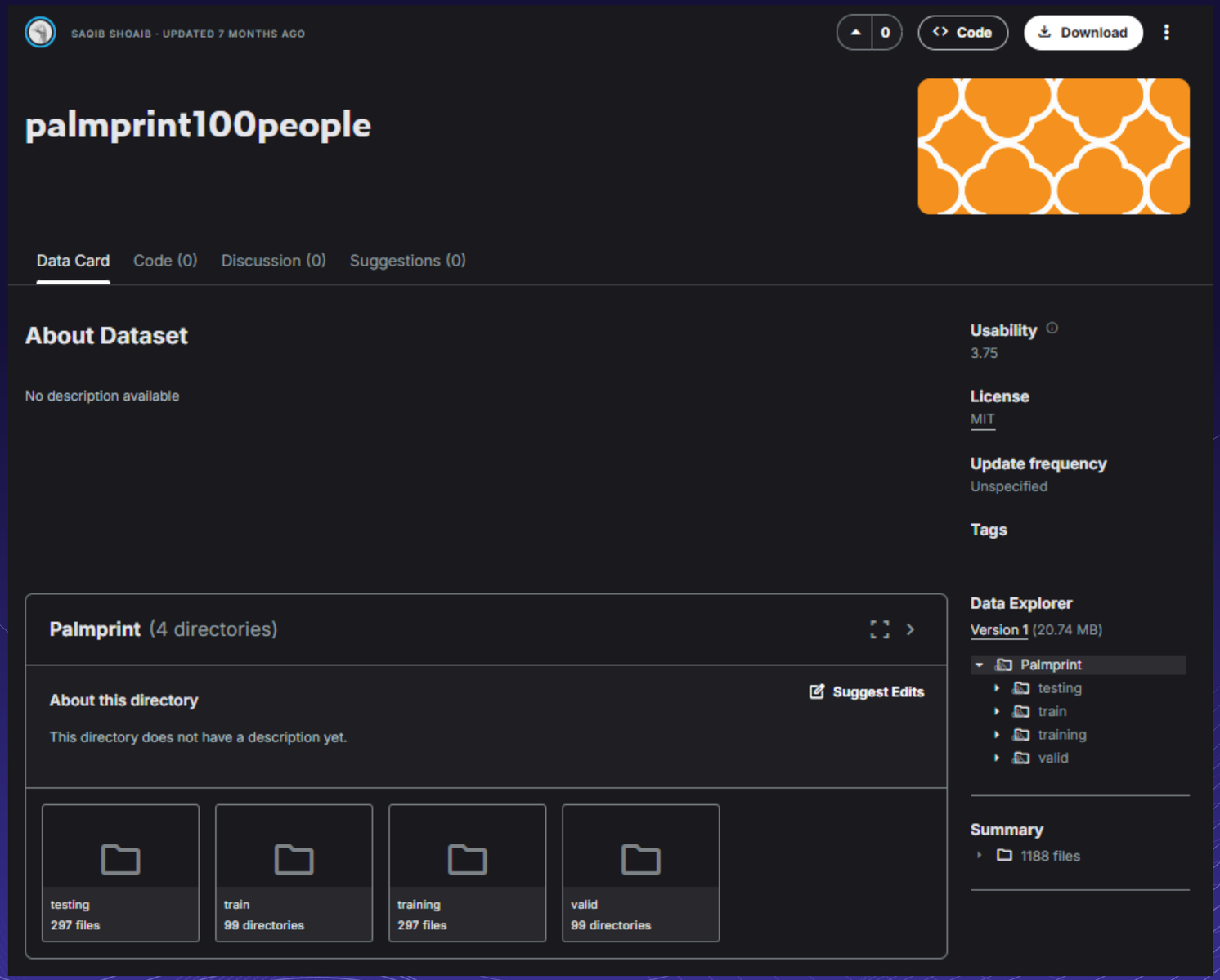
2206131

Rakha Dhifiargo Hariadi

2209489

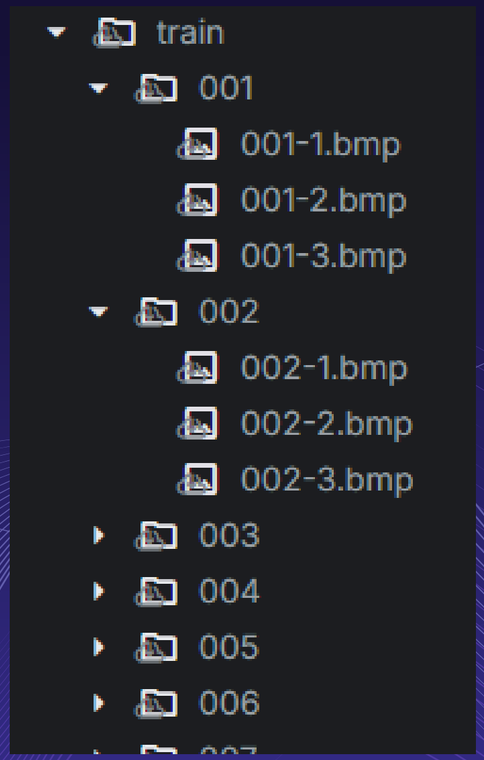
Dataset

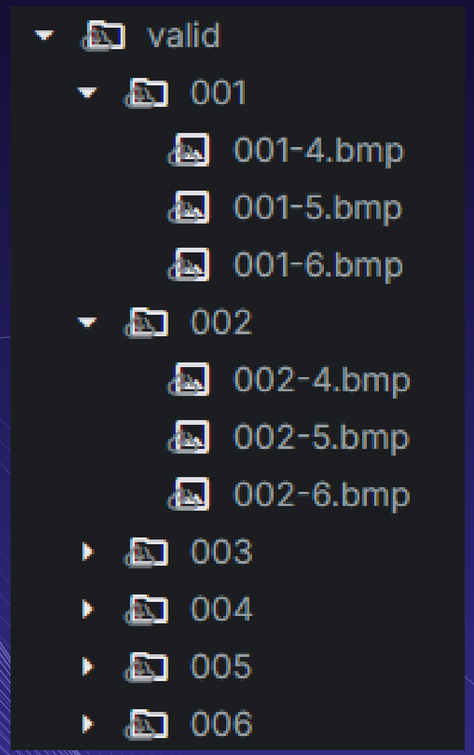
https://www.kaggle.com/datasets/saqibshoaibdz/palmprint100people



Dataset Task 1

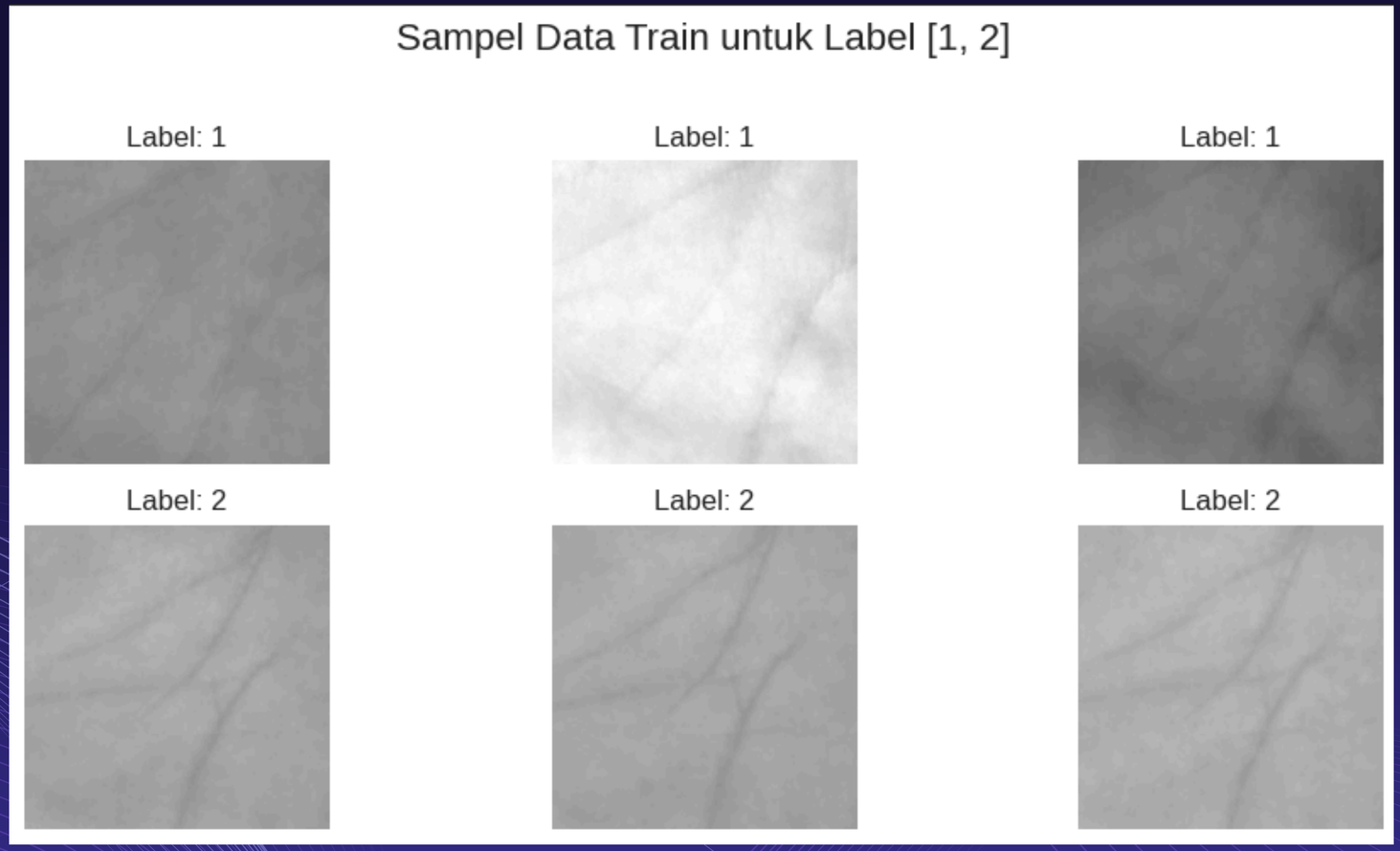
https://www.kaggle.com/datasets/saqibshoaibdz/palmprint100people



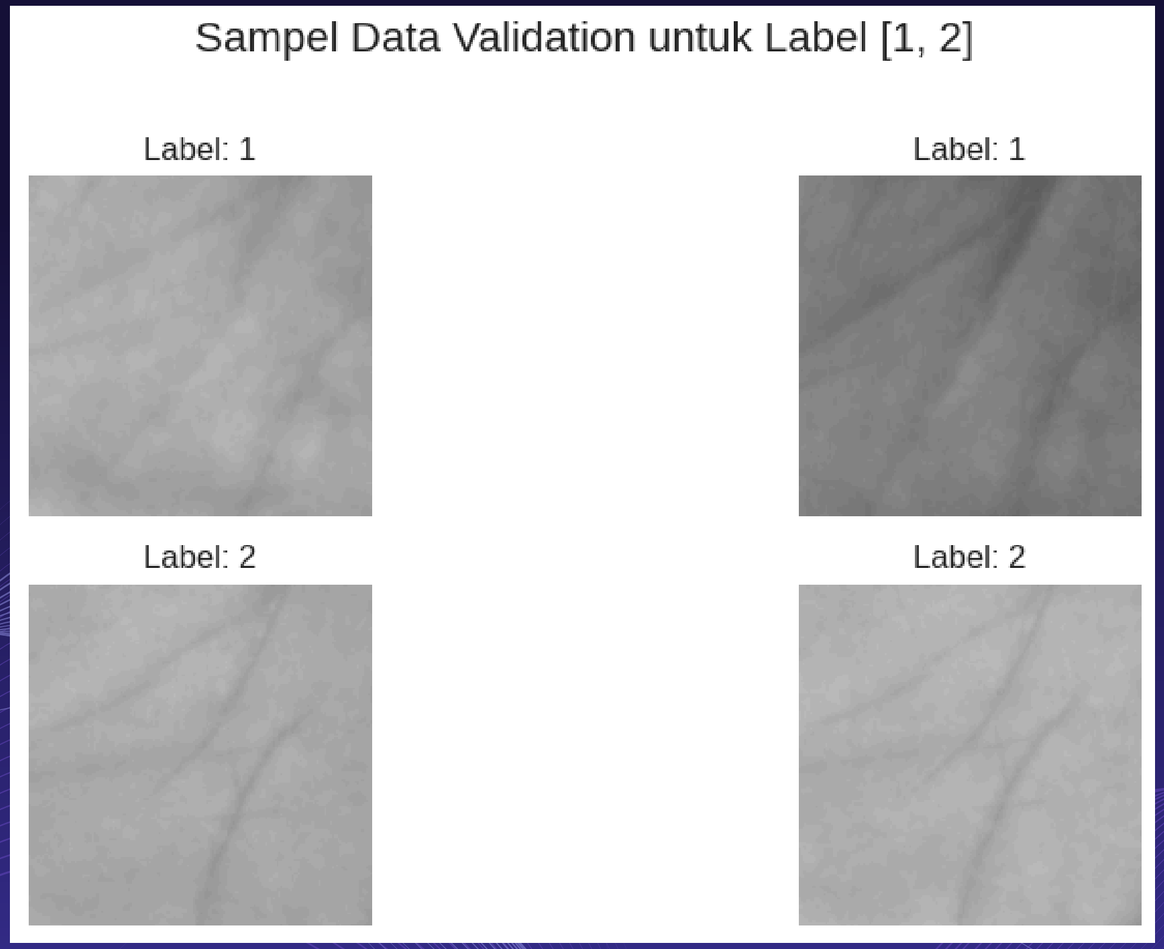


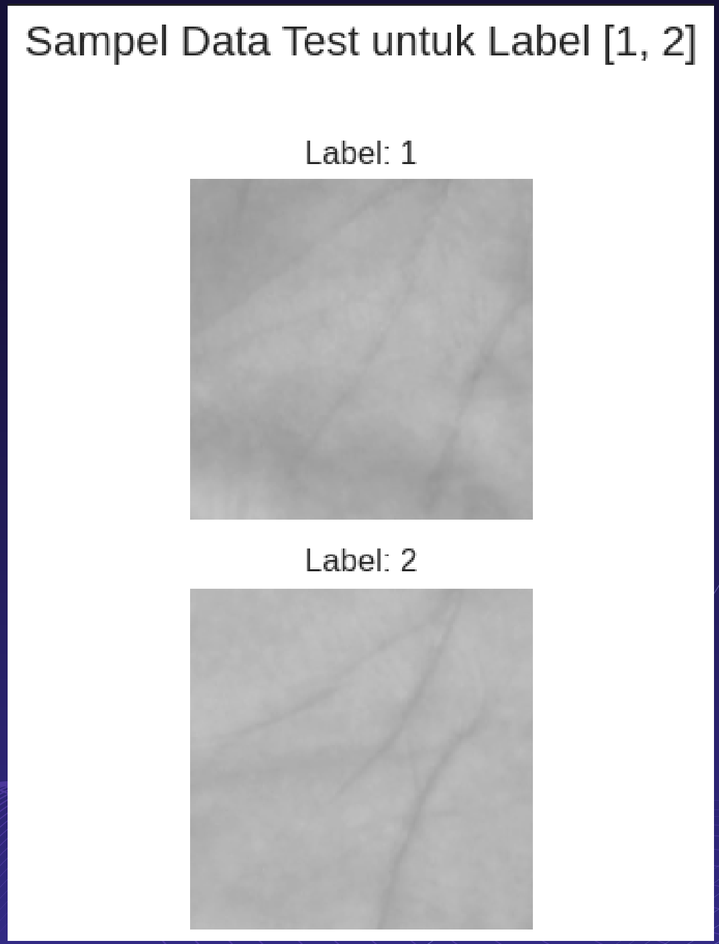

```
Informasi Data Train
Jumlah gambar train: 297
Total Subjek/Label Unik: 99
Shape dari array gambar train: (297, 128, 128, 3)

Informasi Data Validasi
Jumlah gambar validasi: 198
Total Subjek/Label Unik: 99

Informasi Data Test
Jumlah gambar uji: 99
Total Subjek/Label Unik: 99
```

Dataset Task 1



Dataset Task 1





Eksperimen 1 Task 1

Data Preparation

```
# Normalisasi
X_train_norm = X_train.astype('float32') / 255.0
X_val_norm = X_val.astype('float32') / 255.0
X_test_new_norm = X_test_new.astype('float32') / 255.0

# One-Hot Encoding
num_classes = np.max(y_train) + 1
y_train_cat = to_categorical(y_train, num_classes=num_classes)
y_val_cat = to_categorical(y_val, num_classes=num_classes)
y_test_new_cat = to_categorical(y_test_new, num_classes=num_classes)
```

Eksperimen 1 Task 1

Model Preparation

```python
input_shape = X_train_norm.shape[1:]

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(128, activation='relu'),
    # Dropout(0.2),

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

model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten (Flatten)	(None, 57600)	0
dense (Dense)	(None, 128)	7,372,928
dense_1 (Dense)	(None, 100)	12,900

Total params: 7,405,220 (28.25 MB)

Trainable params: 7,405,220 (28.25 MB)

Non-trainable params: 0 (0.00 B)

Eksperimen 1 Task 1

Eksekusi

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(
    X_train_norm,
    y_train_cat,
    epochs=50,
    # batch_size=32,
    validation_data=(X_val_norm, y_val_cat),
)
```

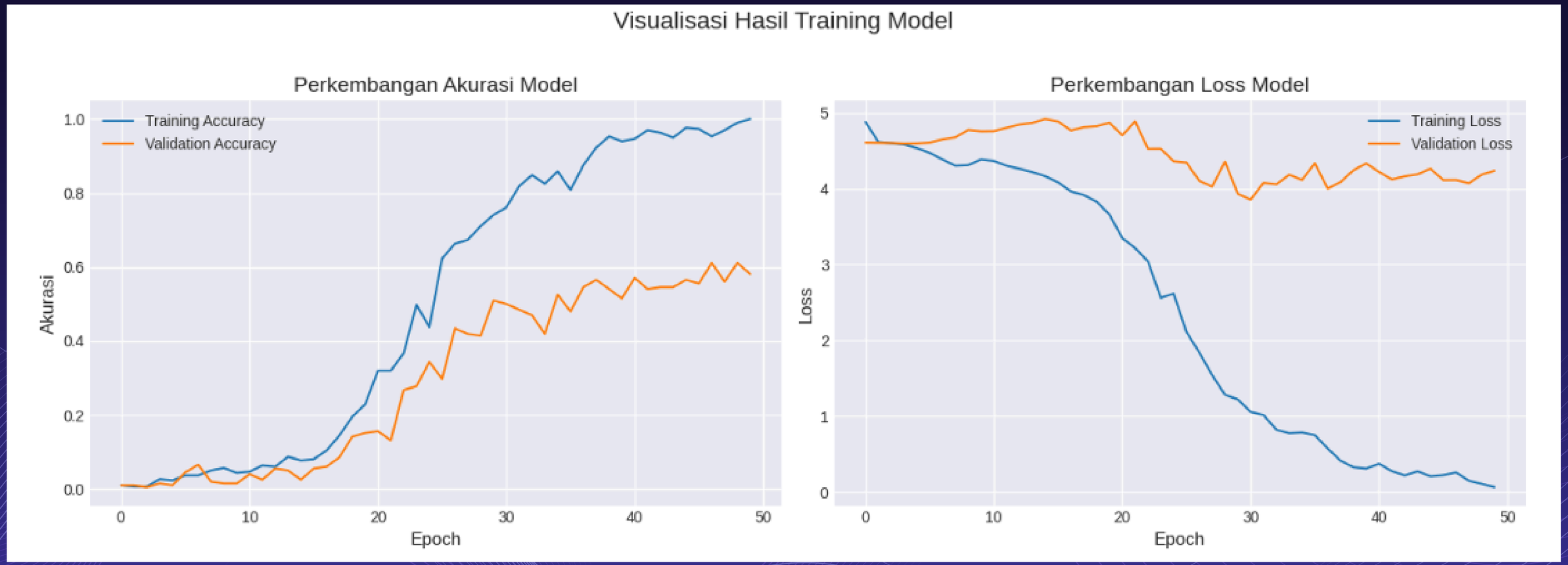
Eksperimen 1 Task 1

Eksekusi

```
Epoch 45/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9781 - loss: 0.2276 - val_accuracy: 0.5657 - val_loss: 4.2639
Epoch 46/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9868 - loss: 0.1617 - val_accuracy: 0.5556 - val_loss: 4.1133
Epoch 47/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9660 - loss: 0.1978 - val_accuracy: 0.6111 - val_loss: 4.1143
Epoch 48/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9777 - loss: 0.1416 - val_accuracy: 0.5606 - val_loss: 4.0724
Epoch 49/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9934 - loss: 0.0973 - val_accuracy: 0.6111 - val_loss: 4.1858
Epoch 50/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 1.0000 - loss: 0.0666 - val_accuracy: 0.5808 - val_loss: 4.2375
```

Eksperimen 1 Task 1

Eksekusi



Eksperimen 1 Task 1

Evaluasi

Subject 98	0.00	0.00	0.00	1
Subject 99	1.00	1.00	1.00	1
accuracy			0.44	99
macro avg	0.35	0.44	0.38	99
weighted avg	0.35	0.44	0.38	99

Eksperimen 2 Task 1

Eksekusi

```
model_2.compile(optimizer='adam',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

history_2 = model_2.fit(
    X_train_norm,
    y_train_cat,
    epochs=100,
    # batch_size=32,
    validation_data=(X_val_norm, y_val_cat),
)
```

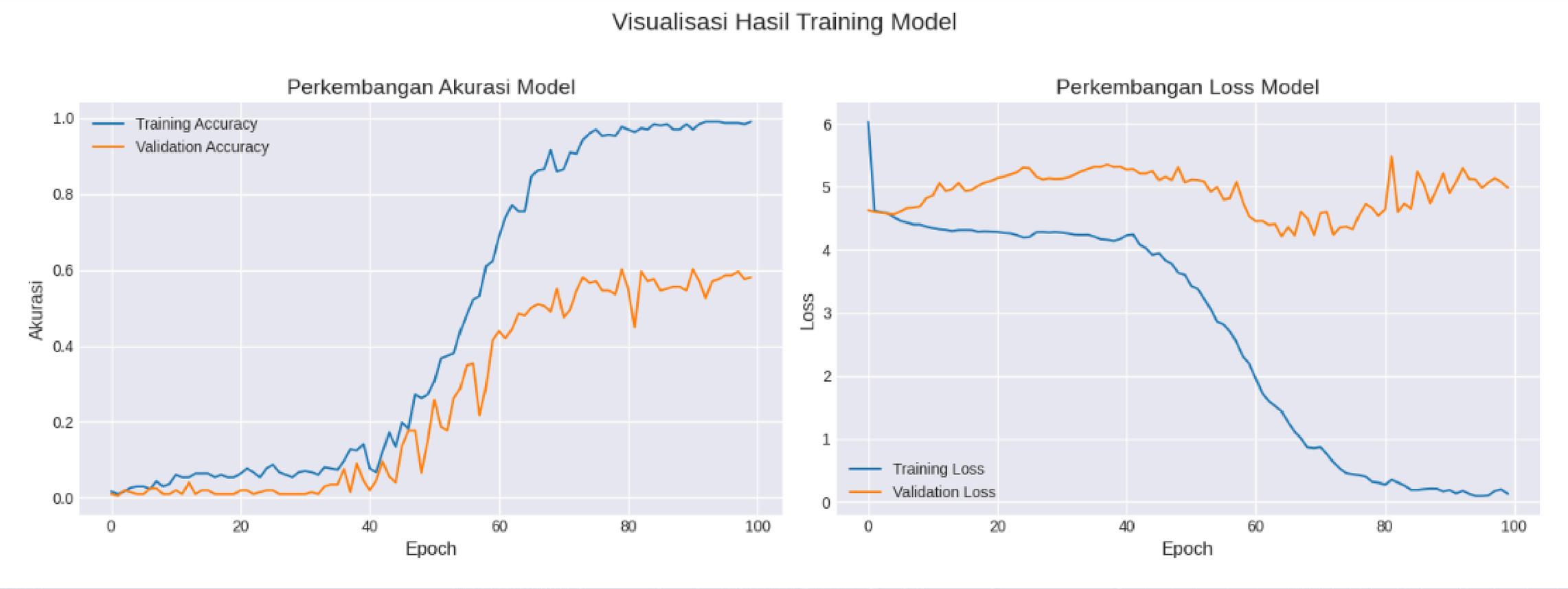
Eksperimen 2 Task 1

Eksekusi

```
Epoch 95/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9918 - loss: 0.0886 - val_accuracy: 0.5758 - val_loss: 5.1181
Epoch 96/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9901 - loss: 0.0804 - val_accuracy: 0.5859 - val_loss: 4.9848
Epoch 97/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9901 - loss: 0.0939 - val_accuracy: 0.5859 - val_loss: 5.0715
Epoch 98/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9914 - loss: 0.1479 - val_accuracy: 0.5960 - val_loss: 5.1386
Epoch 99/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9820 - loss: 0.1583 - val_accuracy: 0.5758 - val_loss: 5.0771
Epoch 100/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9883 - loss: 0.1256 - val_accuracy: 0.5808 - val_loss: 4.9849
```

Eksperimen 2 Task 1

Eksekusi



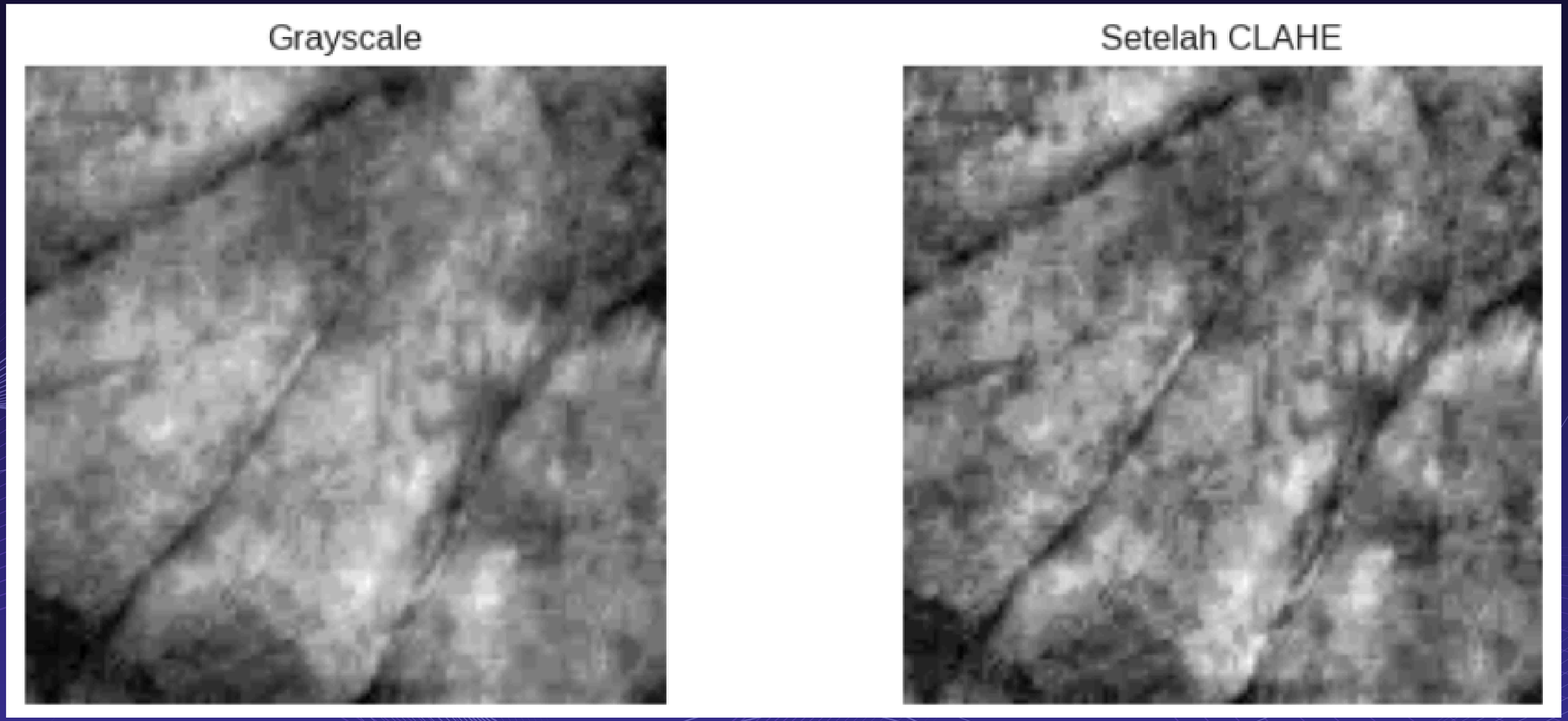
Eksperimen 2 Task 1

Evaluasi

Subject 98	0.00	0.00	0.00	1
Subject 99	0.00	0.00	0.00	1
accuracy			0.47	99
macro avg	0.37	0.47	0.40	99
weighted avg	0.37	0.47	0.40	99

Eksperimen 3 Task 1

Praproses



Eksperimen 3 Task 1

Data preparation

```
X_train_norm = X_train_clahe.astype('float32') / 255.0
X_val_norm = X_val_clahe.astype('float32') / 255.0
X_test_new_norm = X_test_new_clahe.astype('float32') / 255.0

X_train_reshaped = np.expand_dims(X_train_norm, axis=-1)
X_val_reshaped = np.expand_dims(X_val_norm, axis=-1)
X_test_new_reshaped = np.expand_dims(X_test_new_norm, axis=-1)

num_classes = np.max(y_train) + 1
y_train_cat = to_categorical(y_train, num_classes=num_classes)
y_val_cat = to_categorical(y_val, num_classes=num_classes)
y_test_new_cat = to_categorical(y_test_new, num_classes=num_classes)
```

Eksperimen 3 Task 1

Model preparation

```
model_3 = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    # Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

model_3.compile(optimizer='adam',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Eksperimen 3 Task 1

Eksekusi

```
history_3 = model_3.fit(
    X_train_reshaped,
    y_train_cat,
    epochs=50,
    # batch_size=32,
    validation_data=(X_val_reshaped, y_val_cat),
)
```

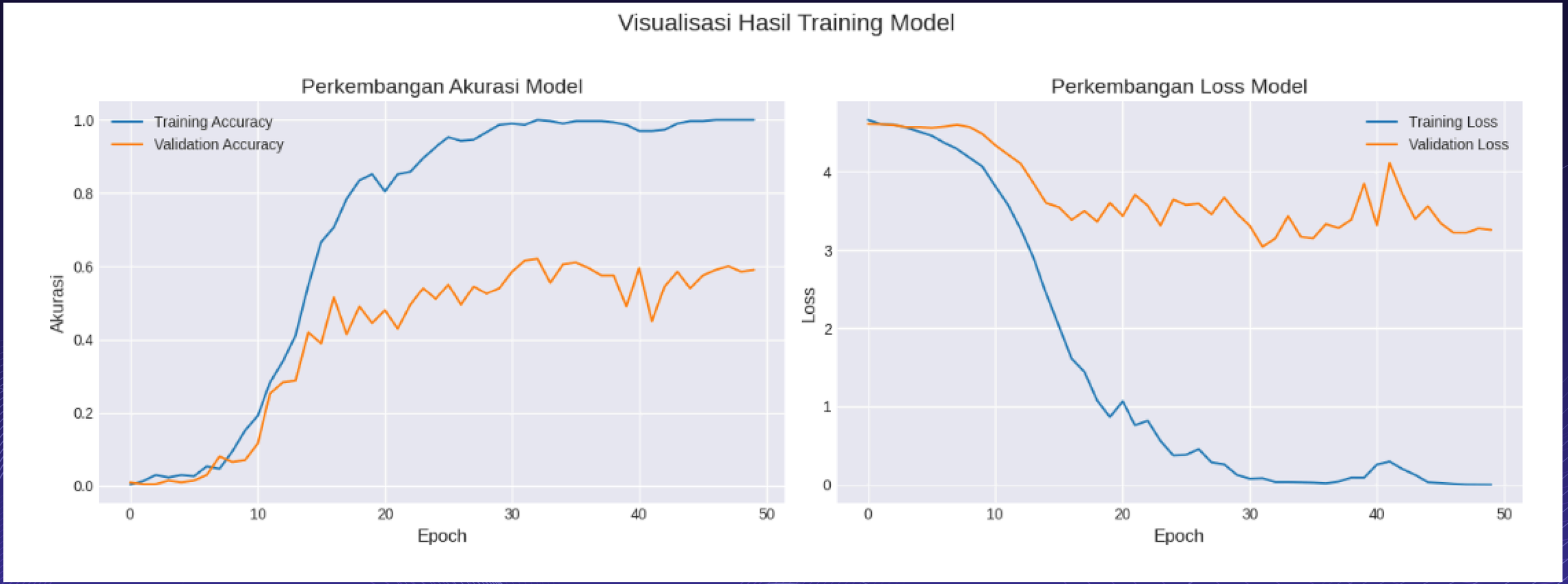
Eksperimen 3 Task 1

Eksekusi

```
Epoch 45/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9914 - loss: 0.0505 - val_accuracy: 0.5404 - val_loss: 3.5584
Epoch 46/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9973 - loss: 0.0263 - val_accuracy: 0.5758 - val_loss: 3.3416
Epoch 47/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.0109 - val_accuracy: 0.5909 - val_loss: 3.2212
Epoch 48/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.0055 - val_accuracy: 0.6010 - val_loss: 3.2195
Epoch 49/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.0060 - val_accuracy: 0.5859 - val_loss: 3.2752
Epoch 50/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.0042 - val_accuracy: 0.5909 - val_loss: 3.2566
```

Eksperimen 3 Task 1

Eksekusi



Eksperimen 3 Task 1

Evaluasi

Subject 98	1.00	1.00	1.00	1
Subject 99	1.00	1.00	1.00	1
accuracy			0.49	99
macro avg	0.40	0.49	0.43	99
weighted avg	0.40	0.49	0.43	99

Hasil sementara

Hasil kurang maksimal.

Jumlah data terbatas, 99 label, tapi per label hanya maksimal 6 data.

Transfer learning atau penggunaan pre-trained models menjadi solusi.

Eksperimen 4 Task 1

Data preparation (MobileNetV3)

```
X_train_rgb = np.stack([X_train_clahe]*3, axis=-1)
X_val_rgb = np.stack([X_val_clahe]*3, axis=-1)
X_test_new_rgb = np.stack([X_test_new_clahe]*3, axis=-1)

X_train_processed = tf.keras.applications.mobilenet_v3.preprocess_input(X_train_rgb)
X_val_processed = tf.keras.applications.mobilenet_v3.preprocess_input(X_val_rgb)
X_test_new_processed = tf.keras.applications.mobilenet_v3.preprocess_input(X_test_new_rgb)

y_train_cat = to_categorical(y_train, num_classes=num_classes)
y_val_cat = to_categorical(y_val, num_classes=num_classes)
y_test_new_cat = to_categorical(y_test_new, num_classes=num_classes)
```

Eksperimen 4 Task 1

Model preparation (MobileNetV3)

```
base_model = tf.keras.applications.MobileNetV3Large(input_shape=(128, 128, 3),
                                                    include_top=False,
                                                    weights='imagenet')

base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
# x = Dropout(0.5)(x)
predictions = Dense(num_classes, activation='softmax')(x)

model_4 = Model(inputs=base_model.input, outputs=predictions)

model_4.compile(optimizer='adam',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Eksperimen 4 Task 1

Model preparation (MobileNetV3)

(Activation)			
global_average_poo… (GlobalAveragePool…	(None, 960)	0	activation_19[0]…
dense_8 (Dense)	(None, 128)	123,008	global_average_p…
dense_9 (Dense)	(None, 100)	12,900	dense_8[0][0]

Total params: 3,132,260 (11.95 MB)

Trainable params: 135,908 (530.89 KB)

Non-trainable params: 2,996,352 (11.43 MB)

Eksperimen 4 Task 1

Eksekusi

```
history_4 = model_4.fit(
    X_train_processed,
    y_train_cat,
    epochs=25,
    # batch_size=32,
    validation_data=(X_val_processed, y_val_cat),
)
```

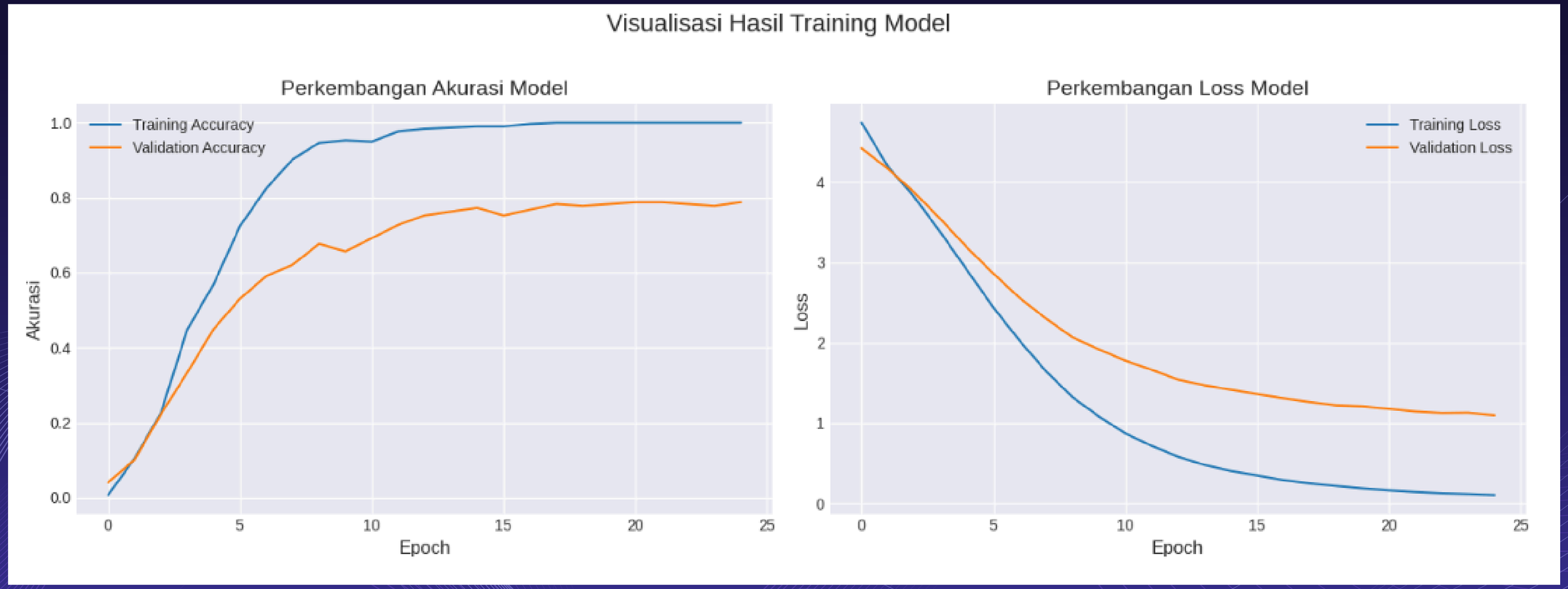
Eksperimen 4 Task 1

Eksekusi

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.2084 - val_accuracy: 0.7778 - val_loss: 1.2197
Epoch 20/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.1839 - val_accuracy: 0.7828 - val_loss: 1.2064
Epoch 21/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 1.0000 - loss: 0.1757 - val_accuracy: 0.7879 - val_loss: 1.1761
Epoch 22/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.1402 - val_accuracy: 0.7879 - val_loss: 1.1423
Epoch 23/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.1143 - val_accuracy: 0.7828 - val_loss: 1.1230
Epoch 24/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 1.0000 - loss: 0.1091 - val_accuracy: 0.7778 - val_loss: 1.1267
Epoch 25/25
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 1.0000 - loss: 0.1084 - val_accuracy: 0.7879 - val_loss: 1.0944
```

Eksperimen 4 Task 1

Eksekusi



Eksperimen 4 Task 1

Evaluasi

Subject 98	1.00	1.00	1.00	1
Subject 99	1.00	1.00	1.00	1
accuracy			0.71	99
macro avg	0.61	0.71	0.64	99
weighted avg	0.61	0.71	0.64	99

Kesimpulan

Eksperimen	Arsitektur	Epochs	Preprocessing	Akurasi Test	Parameter
1	CNN Sederhana	50	RGB Normalisasi	44%	7.4M
2	CNN Sederhana	100	RGB Normalisasi	47%	7.4M
3	CNN Sederhana	50	Grayscale + CLAHE	49%	7.4M
4	MobileNetV3	25	Grayscale + CLAHE + RGB	**71%**	3.1M

Kesimpulan

Pada kasus dataset ini, data terlalu sedikit untuk dilakukan training dengan metode biasa, sehingga harus memanfaatkan pre-trained model..

Penggunaan MobileNetV3 menjadi pilihan dikarenakan asumsi jika nanti inferensi menggunakan perangkat mobile.

Dan jika dibandingkan melalui efisiensi, jelas metode transfer learning merupakan pilihan terbaik mengingat waktu training hampir sama dengan metode konvensional dengan akurasi yang sangat jauh.

Rekomendasi

- Data Augmentation
- Regularization
- Fine-tuning

Thank You
for Your Attention