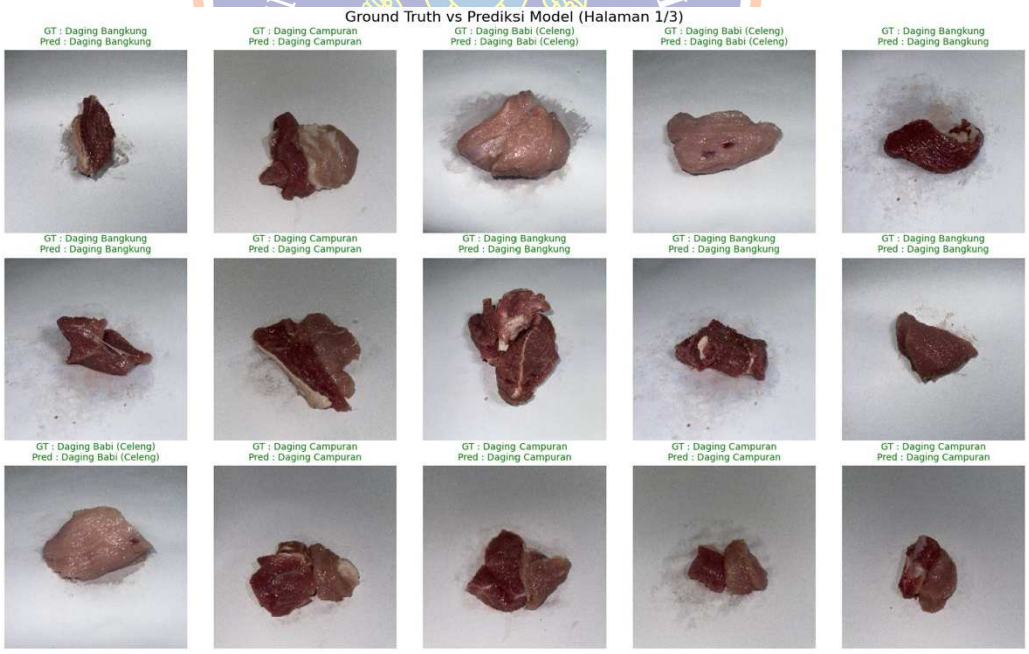
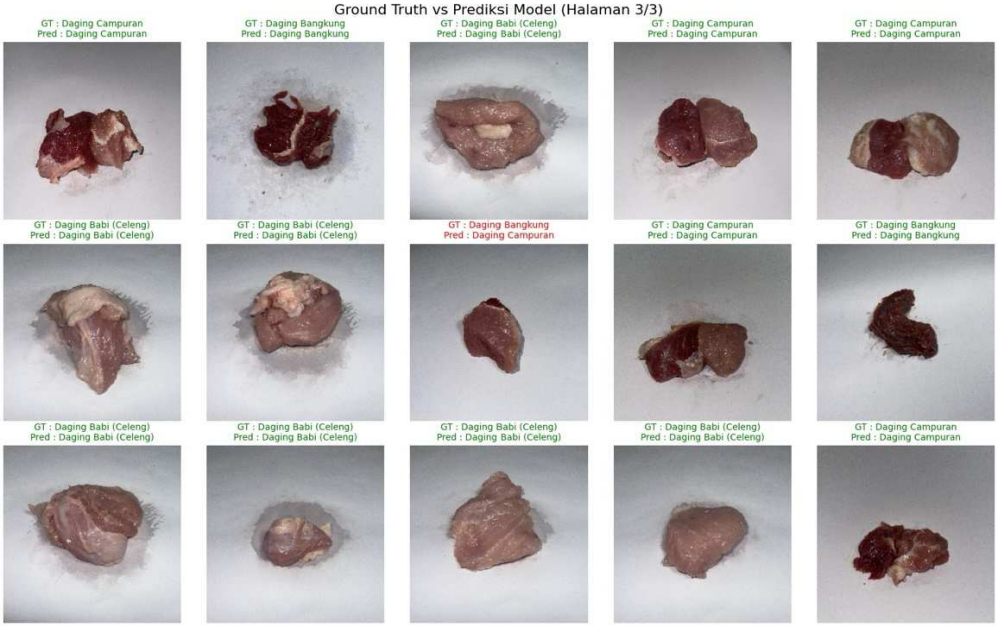
LAMPIRAN

Lampiran 1. Wawancara ke Dinas Pertanian dan Peternakan Kabupaten Buleleng

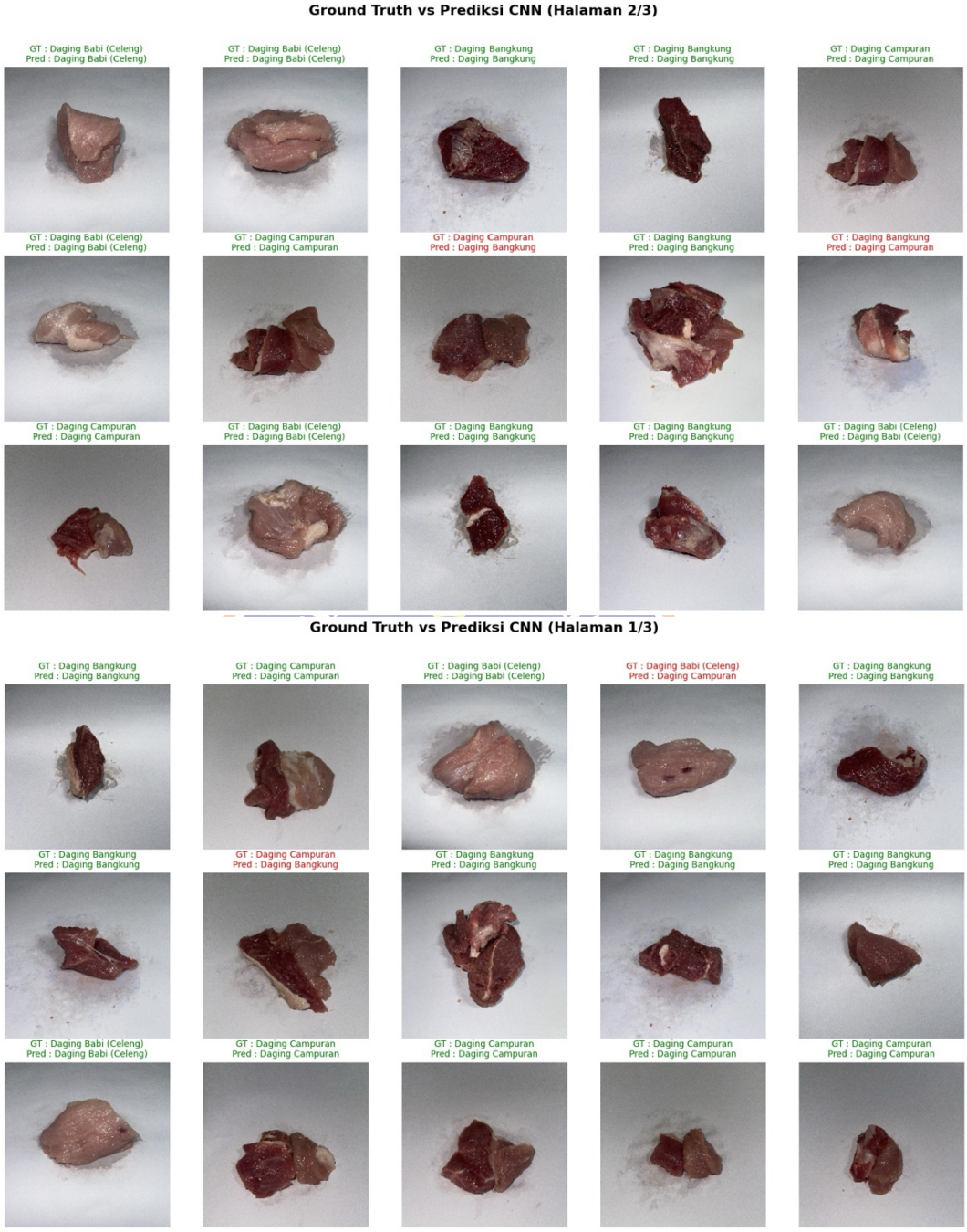
Lampiran 2. Ground Truth vs Prediksi Model VGG16

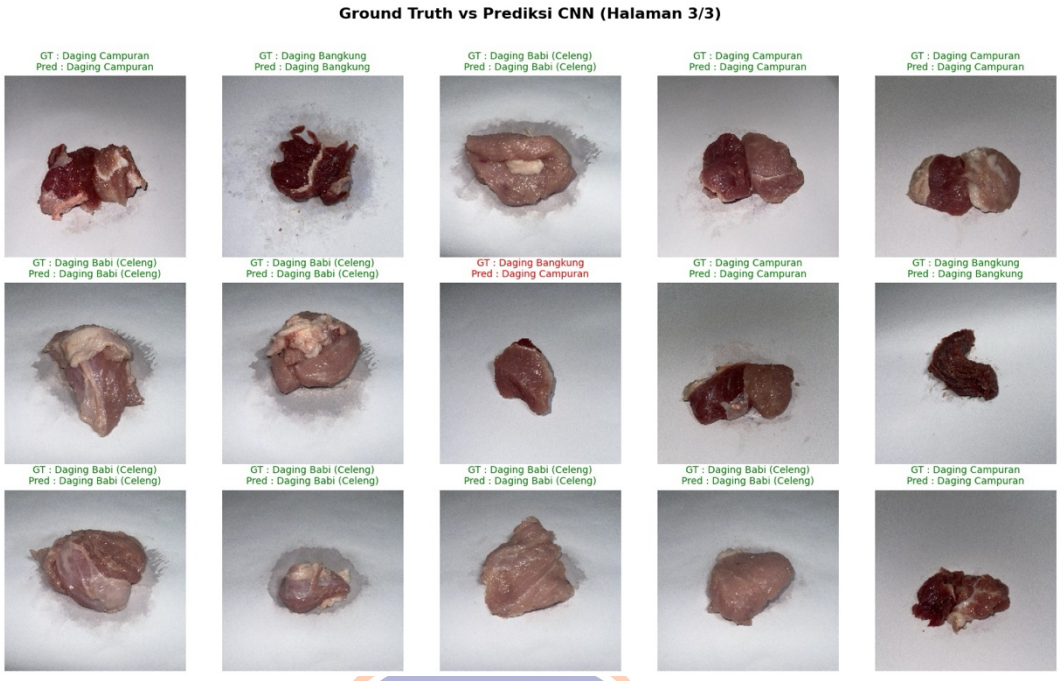



Ground Truth vs Prediksi Model (Halaman 2/3)
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Bangkung
Pred : Daging Bangkung
GT : Daging Bangkung
Pred : Daging Bangkung
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Bangkung
Pred : Daging Bangkung
GT : Daging Bangkung
Pred : Daging Babi (Celeng)
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Bangkung
Pred : Daging Bangkung
GT : Daging Bangkung
Pred : Daging Bangkung
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)


Ground Truth vs Prediksi Model (Halaman 3/3)
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Bangkung
Pred : Daging Bangkung
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Bangkung
Pred : Daging Campuran
GT : Daging Campuran
Pred : Daging Campuran
GT : Daging Bangkung
Pred : Daging Bangkung
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Babi (Celeng)
Pred : Daging Babi (Celeng)
GT : Daging Campuran
Pred : Daging Campuran

Lampiran 3. Ground Truth vs Prediksi Model CNN (Convolutional Neural Network)





Lampiran 4. Code Arsitektur VGG16

```
import tensorflow as tf
from tensorflow.keras.layers import (
 Conv2D, MaxPooling2D, GlobalAveragePooling2D,
GlobalMaxPooling2D,
 BatchNormalization, Dense, Dropout, Concatenate, Input,
Activation
)
from tensorflow.keras.models import Model
from tensorflow.keras.applications import VGG16
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

# ============================================================
# 1. PERSIAPAN VARIABEL
# ============================================================
class_count = len(train_gen.class_indices)
img_size = (224, 224)
img_shape = (img_size[0], img_size[1], 3)

print(f"Jumlah kelas: {class_count}")

# ============================================================
# 2. LOAD VGG16 (PRETRAINED)
# ============================================================
vgg_base = VGG16(
 include_top=False,
 weights='imagenet',
 input_shape=img_shape
)

# Freeze
vgg_base.trainable = False
```

```
for layer in vgg_base.layers[-2:]:
 layer.trainable = True

def build_cnn_branch(input_layer):
 x = Conv2D(32, (3,3), padding='same',
kernel_initializer='he_normal')(input_layer)
 x = BatchNormalization()(x)
 x = Activation('relu')(x)
 x = MaxPooling2D(2)(x)

 x = Conv2D(64, (3,3), padding='same',
kernel_initializer='he_normal')(x)
 x = BatchNormalization()(x)
 x = Activation('relu')(x)

 return GlobalAveragePooling2D()(x)

# ============================================================
# 3. ARSITEKTUR MODEL
# ============================================================
input_layer = Input(shape=img_shape)

x_vgg = vgg_base(input_layer)
x_vgg = GlobalAveragePooling2D()(x_vgg)
x_vgg = BatchNormalization()(x_vgg)

merged = Concatenate()([x_vgg, x_cnn])

merged = Dense(512, kernel_initializer='he_normal')(merged)
merged = BatchNormalization()(merged)
merged = Activation('relu')(merged)
merged = Dropout(0.7)(merged)

merged = Dense(256, kernel_initializer='he_normal')(merged)
merged = BatchNormalization()(merged)
merged = Activation('relu')(merged)
merged = Dropout(0.5)(merged)

output_layer = Dense(class_count, activation='softmax')(merged)

model = Model(inputs=input_layer, outputs=output_layer)

# ============================================================
# 4. COMPILE (Parameter)
# ============================================================
model.compile(
 optimizer=Adam(learning_rate=0.0001),
 loss='categorical_crossentropy',
 metrics=['accuracy']
)

model.summary()

# ============================================================
# 5. CALLBACKS
# ============================================================
early_stopping = EarlyStopping(
 monitor='val_loss',
```

```
 patience=8,
 restore_best_weights=True,
 verbose=1
)

# ============================================================
# 6. TRAINING
# ============================================================
print("\n--- Memulai Training VGG16 ---")
history = model.fit(
 train_gen,
 validation_data=val_gen,
 epochs=30,
 callbacks=[early_stopping],
 verbose=1
)
```

Lampiran 5. Code Arsitektur *CNN (Convolutional Neural Network)*

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.layers import (
 Conv2D, MaxPooling2D, Flatten, BatchNormalization,
 Dense, Dropout, Input, Activation
)
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping,
ReduceLROnPlateau

# ============================================================
# 1. DATA AUGMENTATION
# ============================================================
train_datagen = ImageDataGenerator(
 rescale=1./255,
 rotation_range=10,
 horizontal_flip=True,
 vertical_flip=False,
 fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# ============================================================
# 2. CNN MURNI
# ============================================================
def build_simple_cnn(img_shape, class_count):
 input_layer = Input(shape=img_shape)

 # Block 1
 x = Conv2D(16, (3, 3), padding='same',
activation='relu')(input_layer)
 x = MaxPooling2D(2, 2)(x)

 # Block 2
```

```
 x = Conv2D(32, (3, 3), padding='same', activation='relu')(x)
 x = MaxPooling2D(2, 2)(x)

 # Block 3
 x = Conv2D(64, (3, 3), padding='same', activation='relu')(x)
 x = MaxPooling2D(2, 2)(x)

 x = Flatten()(x)

 # Dense Layer
 x = Dense(64, activation='relu')(x)
 x = Dropout(0.2)(x) # Dropout kecil saja

 output_layer = Dense(class_count, activation='softmax')(x)
 return Model(inputs=input_layer, outputs=output_layer)

model = build_simple_cnn((224, 224, 3), 3)

# ============================================================
# 3. COMPILE
# ============================================================
model.compile(
 optimizer=Adam(learning_rate=0.0001),
 loss='categorical_crossentropy',
 metrics=['accuracy']
)

# ============================================================
# 4. EARLY STOPPING
# ============================================================
early_stop = EarlyStopping(
 monitor='val_loss',
 patience=5,
 restore_best_weights=True,
 verbose=1
)

# ============================================================
# 5. TRAINING
# ============================================================
history = model.fit(
 train_gen,
 validation_data=val_gen,
 epochs=30,
 verbose=1
)
```