```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import cv2
from google.colab.patches import cv2_imshow
from pathlib import Path
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import load_model, Sequential
from tensorflow.keras.preprocessing.image import img_to_array, ImageDataGenerator
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, GlobalAveragePooling2D, Dense, Flatten, Dropout
from tensorflow.keras.applications.vgg16 import VGG16
from sklearn.metrics import confusion_matrix, classification_report


train_val_dir = Path('/content/drive/MyDrive/AI_Capstone_2023/first_dataset')


filepaths = list(train_val_dir.glob(r'**/*.JPG'))
labels = list(map(lambda x: os.path.split(os.path.split(os.path.split(x)[0])[0])[1], filepaths))

# Sort filepaths and labels together
filepaths, labels = zip(*sorted(zip(filepaths, labels)))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)
labels = pd.Series(labels, name='Label')

image_df = pd.concat([filepaths, labels], axis=1)


image_df


train_df, test_df = train_test_split(image_df, train_size=0.7, shuffle=True, random_state=1)


train_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
    width_shift_range=0.2,
    height_shift_range=0.2,
    validation_split=0.2
)

test_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255
)


train_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='binary',
    batch_size=32,
    shuffle=True,
    seed=7,
    subset='training'
)

val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='binary',
```

```
    batch_size=32,
    shuffle=True,
    seed=7,
    subset='validation'
)

test_images = test_generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='binary',
    batch_size=32,
    shuffle=False
)
```

```
Found 1848 validated image filenames belonging to 2 classes.
Found 462 validated image filenames belonging to 2 classes.
Found 991 validated image filenames belonging to 2 classes.
```

```
input_tensor = tf.keras.Input(shape=(224, 224, 3))
base_model = VGG16(
    weights = 'imagenet',
    include_top = False,
    input_tensor=input_tensor
)

for layer in base_model.layers:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
58889256/58889256 [==============================] - 3s 0us/step
```

```
model = Sequential([
    base_model,
    Flatten(),
    Dense(1, activation='sigmoid')
])

model.summary()
```

```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['binary_accuracy']
)
```

```
history = model.fit(
    train_images,
    validation_data=val_images,
    epochs=100,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=5,
            restore_best_weights=True
        ),
        ModelCheckpoint(
            '/content/drive/MyDrive/AI_Capstone_2023/V7_model.h5',
            save_best_only=True, monitor='val_loss', mode='min'
        ),
        tf.keras.callbacks.ReduceLROnPlateau(
            monitor='val_loss',
            patience=3
        )
    ]
)
```

```
Epoch 1/100
58/58 [==============================] - 866s 15s/step - loss: 0.3092 - binary_accuracy: 0.8615 - val_loss: 0.1566 - val_binary_accuracy
Epoch 2/100
58/58 [==============================] - 113s 2s/step - loss: 0.1268 - binary_accuracy: 0.9632 - val_loss: 0.0902 - val_binary_accuracy:
Epoch 3/100
58/58 [==============================] - 114s 2s/step - loss: 0.0761 - binary_accuracy: 0.9876 - val_loss: 0.0583 - val_binary_accuracy:
Epoch 4/100
58/58 [==============================] - 114s 2s/step - loss: 0.0560 - binary_accuracy: 0.9924 - val_loss: 0.0513 - val_binary_accuracy:
Epoch 5/100
58/58 [==============================] - 113s 2s/step - loss: 0.0445 - binary_accuracy: 0.9962 - val_loss: 0.0407 - val_binary_accuracy:
Epoch 6/100
58/58 [==============================] - 113s 2s/step - loss: 0.0349 - binary_accuracy: 0.9968 - val_loss: 0.0350 - val_binary_accuracy:
Epoch 7/100
58/58 [==============================] - 111s 2s/step - loss: 0.0309 - binary_accuracy: 0.9951 - val_loss: 0.0292 - val_binary_accuracy:
Epoch 8/100
58/58 [==============================] - 110s 2s/step - loss: 0.0258 - binary_accuracy: 0.9978 - val_loss: 0.0226 - val_binary_accuracy:
Epoch 9/100
58/58 [==============================] - 112s 2s/step - loss: 0.0252 - binary_accuracy: 0.9978 - val_loss: 0.0235 - val_binary_accuracy:
Epoch 10/100
58/58 [==============================] - 110s 2s/step - loss: 0.0207 - binary_accuracy: 0.9968 - val_loss: 0.0192 - val_binary_accuracy:
Epoch 11/100
58/58 [==============================] - 110s 2s/step - loss: 0.0173 - binary_accuracy: 0.9989 - val_loss: 0.0204 - val_binary_accuracy:
Epoch 12/100
58/58 [==============================] - 110s 2s/step - loss: 0.0135 - binary_accuracy: 1.0000 - val_loss: 0.0200 - val_binary_accuracy:
Epoch 13/100
58/58 [==============================] - 110s 2s/step - loss: 0.0153 - binary_accuracy: 0.9978 - val_loss: 0.0271 - val_binary_accuracy:
Epoch 14/100
58/58 [==============================] - 112s 2s/step - loss: 0.0143 - binary_accuracy: 0.9989 - val_loss: 0.0140 - val_binary_accuracy:
Epoch 15/100
58/58 [==============================] - 110s 2s/step - loss: 0.0119 - binary_accuracy: 0.9995 - val_loss: 0.0138 - val_binary_accuracy:
Epoch 16/100
58/58 [==============================] - 109s 2s/step - loss: 0.0104 - binary_accuracy: 1.0000 - val_loss: 0.0114 - val_binary_accuracy:
Epoch 17/100
58/58 [==============================] - 110s 2s/step - loss: 0.0132 - binary_accuracy: 0.9989 - val_loss: 0.0102 - val_binary_accuracy:
Epoch 18/100
58/58 [==============================] - 109s 2s/step - loss: 0.0125 - binary_accuracy: 0.9989 - val_loss: 0.0132 - val_binary_accuracy:
Epoch 19/100
58/58 [==============================] - 110s 2s/step - loss: 0.0110 - binary_accuracy: 0.9984 - val_loss: 0.0153 - val_binary_accuracy:
Epoch 20/100
58/58 [==============================] - 109s 2s/step - loss: 0.0106 - binary_accuracy: 1.0000 - val_loss: 0.0135 - val_binary_accuracy:
Epoch 21/100
 4/58 [=>............................] - ETA: 1:19 - loss: 0.0068 - binary_accuracy: 1.0000
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-12-ecafe67887ae> in <cell line: 1>()
----> 1 history = model.fit(
      2     train_images,
      3     validation_data=val_images,
      4     epochs=100,
      5     callbacks=[

                                 8 frames
/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx,
name)
     50   try:
     51     ctx.ensure_initialized()
---> 52     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     53                                         inputs, attrs, num_outputs)
     54   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

```
missouri_test_dir = '/content/drive/MyDrive/AI_Capstone_2023/missouri_test_set'
personal_test_dir = '/content/drive/MyDrive/AI_Capstone_2023/personal_test_dataset'

alt_test_dir = missouri_test_dir

alt_test_images = test_generator.flow_from_directory(
    directory=alt_test_dir,
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='binary',
    batch_size=32,
    shuffle=False
)
```

```
Found 400 images belonging to 2 classes.
```

```
model = load_model('/content/drive/MyDrive/AI_Capstone_2023/V7_model.h5')


#images_to_test = test_images
images_to_test = alt_test_images

results = model.evaluate(images_to_test, verbose=1)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
13/13 [==============================] - 271s 22s/step - loss: 0.0832 - binary_accuracy: 0.9675
    Test Loss: 0.08320
Test Accuracy: 96.75%
```

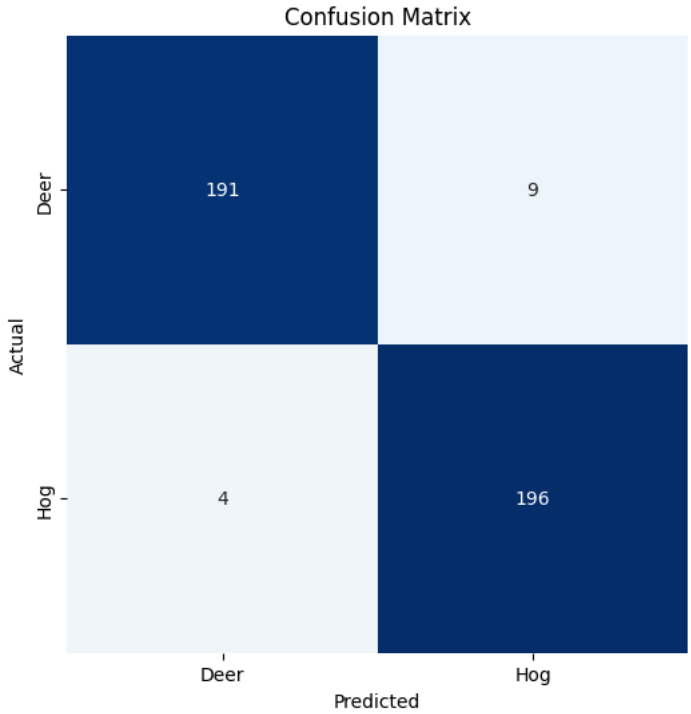
```
predictions = (model.predict(images_to_test) >= 0.5).astype(int)

cm = confusion_matrix(images_to_test.labels, predictions, labels=[0, 1])
clr = classification_report(images_to_test.labels, predictions, labels=[0, 1], target_names=["Deer", "Hog"])

plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='g', vmin=0, cmap='Blues', cbar=False)
plt.xticks(ticks=[0.5, 1.5], labels=["Deer", "Hog"])
plt.yticks(ticks=[0.5, 1.5], labels=["Deer", "Hog"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

print("Classification Report:\n----------------------\n", clr)
```

```
13/13 [==============================] - 15s 1s/step
```




```
Classification Report:
----------------------
              precision    recall  f1-score   support

        Deer       0.98      0.95      0.97       200
         Hog       0.96      0.98      0.97       200

    accuracy                           0.97       400
   macro avg       0.97      0.97      0.97       400
weighted avg       0.97      0.97      0.97       400
```

Class prediction on single images

```
model = load_model('/content/drive/MyDrive/AI_Capstone_2023/V7_model.h5')


image_path = '/content/drive/MyDrive/AI_Capstone_2023/pipeline_end_test/deer2.jpg'
input_image = cv2.imread(image_path, cv2.IMREAD_COLOR)
height, width, _ = input_image.shape
#print(height, width)
original_scale = (width, height)

piped_image = cv2.resize(input_image, (224, 224))


piped_image_array = img_to_array(piped_image)
piped_image_array = np.expand_dims(piped_image_array, axis=0)

# Predict the class of the input image
prediction = model.predict(piped_image_array)
print(prediction)
formated_pred = np.format_float_positional(prediction[0][0], precision=50)
print(formated_pred)
```

```
1/1 [==============================] - 0s 19ms/step
[[0.]]
0.
```

```
threshold = 0.5

# Check if the image belongs to either class
if prediction < threshold:
    class_name = "White-Tailed Deer"
    certainty = 100 - (prediction[0][0] * 100)
elif prediction >= threshold:
    class_name = "Wild Hog"
    certainty = prediction[0][0] * 100

# Display the class name and certainty
text = f"{class_name}: {certainty:.2f}%"
#output_image = cv2.resize(input_image, original_scale)
cv2.rectangle(input_image, (0, 0), (width, 30), (255, 255, 255), -1)
cv2.putText(input_image, text, (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2)
#cv2.rectangle(input_image, (0, 0), (width, height), (0, 255, 0), 2)

cv2_imshow(input_image)
```