```
!pip install -q kaggle
```

```
from google.colab import files
files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
!mkdir ~/.kaggle
```

```
!mv kaggle.json ~/.kaggle/
```

```
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d kaggleashwin/vehicle-type-recognition
```

```
Dataset URL: https://www.kaggle.com/datasets/kaggleashwin/vehicle-type-recognition
License(s): apache-2.0
Downloading vehicle-type-recognition.zip to /content
 96% 153M/159M [00:01<00:00, 143MB/s]
100% 159M/159M [00:01<00:00, 149MB/s]
```

```
!kaggle datasets download kaushalrimal619/lumpy-skin-disease-cow-images
```

```
Dataset URL: https://www.kaggle.com/datasets/kaushalrimal619/lumpy-skin-disease-cow-images
License(s): unknown
Downloading lumpy-skin-disease-cow-images.zip to /content
100% 4.27G/4.28G [00:58<00:00, 85.9MB/s]
100% 4.28G/4.28G [00:58<00:00, 78.4MB/s]
```

```
!unzip lumpy-skin-disease-cow-images.zip
```

```
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

if tf.test.gpu_device_name():
    print(f"Using GPU: {tf.test.gpu_device_name()}")
else:
    print("No GPU found. Make sure you've enabled GPU in Colab.")
```

```
Num GPUs Available:  0
No GPU found. Make sure you've enabled GPU in Colab.
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
import os
import shutil
import matplotlib.pyplot as plt



infected_dir = '/content/infected/infected'
normal_dir = '/content/normal/normal'

def create_dataset(infected_dir, normal_dir):
    dataset_dir = '/content/dataset'
    train_dir = os.path.join(dataset_dir, 'train')
    os.makedirs(train_dir, exist_ok=True)

    infected_train_dir = os.path.join(train_dir, 'infected')
    normal_train_dir = os.path.join(train_dir, 'normal')
    os.makedirs(infected_train_dir, exist_ok=True)
    os.makedirs(normal_train_dir, exist_ok=True)

    for file in os.listdir(infected_dir):
        shutil.copy(os.path.join(infected_dir, file), infected_train_dir)
```

```
    for file in os.listdir(normal_dir):
        shutil.copy(os.path.join(normal_dir, file), normal_train_dir)

    return dataset_dir

dataset_dir = create_dataset(infected_dir, normal_dir)



train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

train_generator = train_datagen.flow_from_directory(
    os.path.join(dataset_dir, 'train'),
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    os.path.join(dataset_dir, 'train'),
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)
```

```
Found 3200 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
```

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),

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

    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # Output layer for binary classification
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Step 4: Compile the Model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


# Step 5: Train the Model
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    train_generator,
    epochs=25,
    validation_data=val_generator,
    callbacks=[early_stop]
)
```

```
Epoch 1/25
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset` class
  self._warn_if_super_not_called()
100/100 ━━━━━━━━━━━━━━━━━━━━ 157s 1s/step - accuracy: 0.8976 - loss: 1.5282 - val_accuracy: 0.5000 - val_loss: 70.8111
Epoch 2/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 132s 1s/step - accuracy: 0.9831 - loss: 0.1825 - val_accuracy: 0.5000 - val_loss: 77.4527
Epoch 3/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 134s 1s/step - accuracy: 0.9876 - loss: 0.2018 - val_accuracy: 0.5038 - val_loss: 29.1804
Epoch 4/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 149s 1s/step - accuracy: 0.9862 - loss: 0.1574 - val_accuracy: 0.5487 - val_loss: 35.5958
Epoch 5/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 124s 1s/step - accuracy: 0.9926 - loss: 0.0601 - val_accuracy: 0.7275 - val_loss: 11.4234
Epoch 6/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 140s 1s/step - accuracy: 0.9950 - loss: 0.0513 - val_accuracy: 0.7400 - val_loss: 11.6773
Epoch 7/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 132s 1s/step - accuracy: 0.9908 - loss: 0.0958 - val_accuracy: 0.8425 - val_loss: 3.9482
Epoch 8/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 124s 1s/step - accuracy: 0.9939 - loss: 0.0628 - val_accuracy: 0.7850 - val_loss: 8.9398
Epoch 9/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 140s 1s/step - accuracy: 0.9911 - loss: 0.1514 - val_accuracy: 0.8562 - val_loss: 6.1548
Epoch 10/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 142s 1s/step - accuracy: 0.9947 - loss: 0.0572 - val_accuracy: 0.6775 - val_loss: 20.4675
Epoch 11/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 132s 1s/step - accuracy: 0.9920 - loss: 0.0654 - val_accuracy: 0.7937 - val_loss: 7.1895
Epoch 12/25
100/100 ━━━━━━━━━━━━━━━━━━━━ 133s 1s/step - accuracy: 0.9988 - loss: 0.0072 - val_accuracy: 0.8388 - val_loss: 4.7486
```

```
# Step 6: Save the Model
model.save('cow_health_classifier.h5')
```

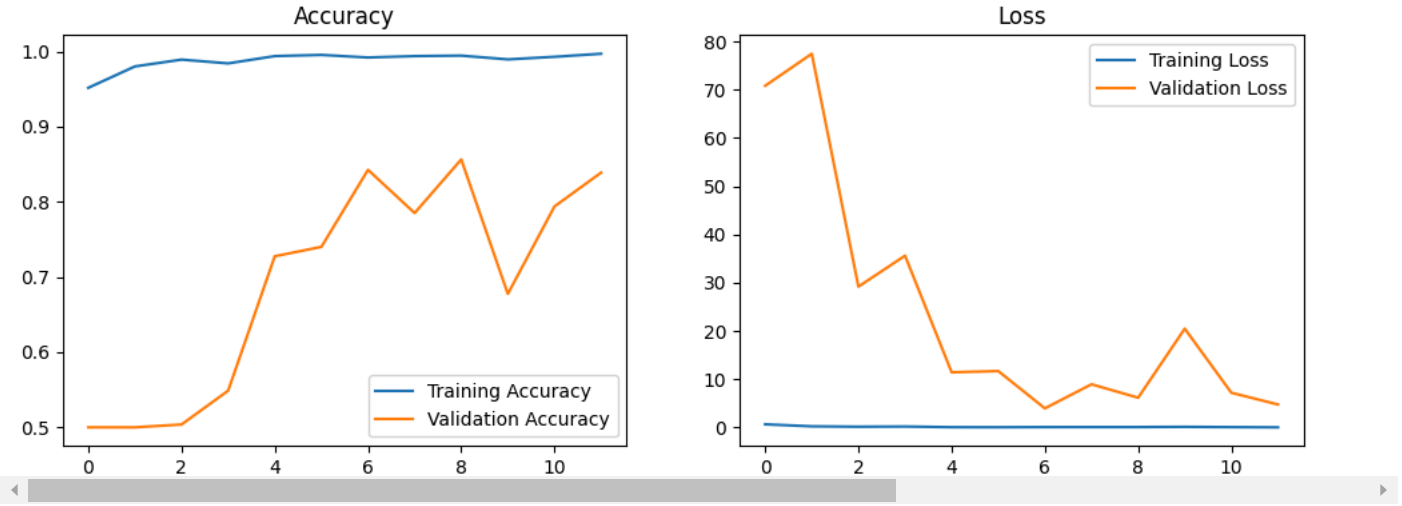
```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consi
```

```
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.show()
```




```
from google.colab import files
files.download('cow_health_classifier.h5')
```

```
from tensorflow.keras.models import load_model
model = load_model('cow_health_classifier.h5')
model.summary()
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you t
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
batch_normalization (BatchNormalization)	(None, 126, 126, 32)	128
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 61, 61, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 28, 28, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 256)	6,422,784
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257

Total params: 6,517,187 (24.86 MB)

```
import matplotlib.pyplot as plt
from PIL import Image


def display_image(image_path):

    img = Image.open(image_path)
    plt.imshow(img)
    plt.axis('off')
    plt.show()


from tensorflow.keras.utils import load_img, img_to_array
import numpy as np


def preprocess_image(image_path):
    img = load_img(image_path, target_size=(128, 128))  # Resize the image to (128, 128)
    img_array = img_to_array(img)  # Convert the image to a numpy array
    img_array = img_array / 255.0  # Normalize the pixel values to [0, 1]
    img_array = np.expand_dims(img_array, axis=0)  # Add a batch dimension
    return img_array


image_path = '/content/unhealthy_1.jpeg'
preprocessed_image = preprocess_image(image_path)
prediction = model.predict(preprocessed_image)


if prediction[0][0] > 0.5:
    print("Unhealthy Cow Detected (Infected)")
else:
    print("Healthy Cow Detected (Normal)")

display_image(image_path)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Unhealthy Cow Detected (Infected)
```

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit