```
!pip install -q tensorflow numpy matplotlib opencv-python
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import zipfile
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Show hidden output

```
zip_path = "archive (1).zip"  # Update if your file has a different name
extract_path = "dataset"

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

os.listdir("dataset/data/train")  # Check extracted folders
```

```
['infected', 'notinfected']
```

```
from PIL import Image

def clean_dataset(folder):
    for subdir, _, files in os.walk(folder):
        for file in files:
            file_path = os.path.join(subdir, file)
            try:
                img = Image.open(file_path)
                img.verify()
                new_name = file.replace(" ", "_").replace("(", "").replace(")", "")
                os.rename(file_path, os.path.join(subdir, new_name))
            except:
                print(f"Removing invalid file: {file_path}")
                os.remove(file_path)

clean_dataset("dataset/data/train")
clean_dataset("dataset/data/test")
```

```
Removing invalid file: dataset/data/test/infected/WhatsApp Image 2022-04-01 at 3.49.45 PM.jpeg
Removing invalid file: dataset/data/test/infected/WhatsApp Image 2022-04-01 at 3.50.20 PM.jpeg
Removing invalid file: dataset/data/test/infected/R.jpg
Removing invalid file: dataset/data/test/infected/WhatsApp Image 2022-04-01 at 3.50.05 PM.jpeg
Removing invalid file: dataset/data/test/infected/WhatsApp Image 2022-04-01 at 3.49.22 PM.jpeg
Removing invalid file: dataset/data/test/infected/OIP.jpg
Removing invalid file: dataset/data/test/notinfected/OIP (1).jpg
Removing invalid file: dataset/data/test/notinfected/img_0_7.jpg
Removing invalid file: dataset/data/test/notinfected/WhatsApp Image 2022-04-01 at 3.39.07 PM.jpeg
Removing invalid file: dataset/data/test/notinfected/WhatsApp Image 2022-04-01 at 3.35.25 PM.jpeg
```

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 32

train_gen = ImageDataGenerator(rescale=1./255, horizontal_flip=True)
test_gen = ImageDataGenerator(rescale=1./255)

train_data = train_gen.flow_from_directory("dataset/data/train", target_size=IMG_SIZE, batch_size=BATCH_SIZE, cl
test_data = test_gen.flow_from_directory("dataset/data/test", target_size=IMG_SIZE, batch_size=BATCH_SIZE, class
```

```
Found 1924 images belonging to 2 classes.
Found 1922 images belonging to 2 classes.
```

```
base_model = tf.keras.applications.MobileNetV2(include_top=False, weights='imagenet', input_shape=(224, 224, 3))
base_model.trainable = False  # Freeze pretrained layers

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Binary classification
])
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['ac

# Train the model
history = model.fit(train_data, validation_data=test_data, epochs=5)
```

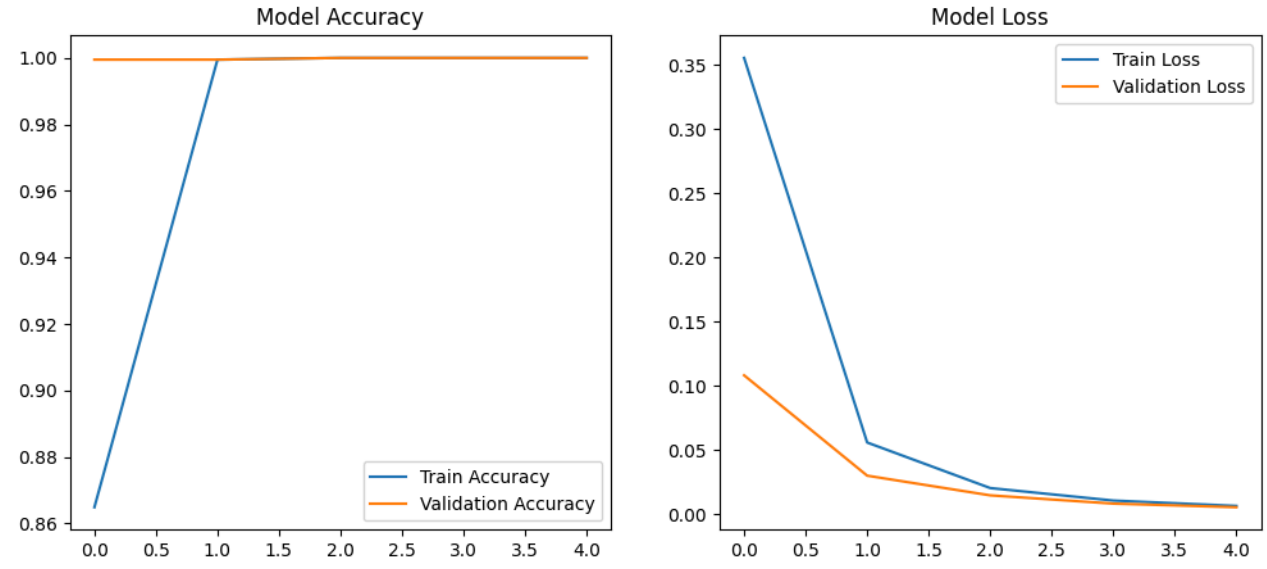
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weig
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning:
  self._warn_if_super_not_called()
Epoch 1/5
61/61 ━━━━━━━━━━━━━━━━━━━━ 184s 3s/step - accuracy: 0.6918 - loss: 0.5457 - val_accuracy: 0.9995 - val_loss: 0.10
Epoch 2/5
61/61 ━━━━━━━━━━━━━━━━━━━━ 185s 3s/step - accuracy: 0.9996 - loss: 0.0776 - val_accuracy: 0.9995 - val_loss: 0.02
Epoch 3/5
61/61 ━━━━━━━━━━━━━━━━━━━━ 179s 3s/step - accuracy: 1.0000 - loss: 0.0239 - val_accuracy: 1.0000 - val_loss: 0.01
Epoch 4/5
61/61 ━━━━━━━━━━━━━━━━━━━━ 179s 3s/step - accuracy: 1.0000 - loss: 0.0114 - val_accuracy: 1.0000 - val_loss: 0.00
Epoch 5/5
61/61 ━━━━━━━━━━━━━━━━━━━━ 177s 3s/step - accuracy: 1.0000 - loss: 0.0071 - val_accuracy: 1.0000 - val_loss: 0.00
```

```
plt.figure(figsize=(12, 5))

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

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

plt.show()
```




```
import random
from tensorflow.keras.preprocessing import image

def show_predictions(num_samples=5):
    fig, axes = plt.subplots(1, num_samples, figsize=(15, 5))

    for ax in axes:
        # Pick a random test image
        img_path = random.choice(test_data.filepaths)
        img = image.load_img(img_path, target_size=(224, 224))
        img_array = image.img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)

        # Get prediction
        pred_prob = model.predict(img_array)[0][0]
        predicted_label = "Infected" if pred_prob > 0.5 else "Not Infected"
```

```
        # Show image & prediction
        ax.imshow(img)
        ax.axis("off")
        ax.set_title(f"Predicted: {predicted_label}\nConfidence: {pred_prob:.2f}")

show_predictions()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step
```


```
import seaborn as sns
from sklearn.metrics import confusion_matrix
import numpy as np

# Get true labels & predictions
true_labels = test_data.classes  # Actual labels
predictions = model.predict(test_data)  # Raw predictions
pred_labels = (predictions > 0.5).astype(int)  # Convert to binary

# Create confusion matrix
cm = confusion_matrix(true_labels, pred_labels)

# Plot
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap="Blues", xticklabels=["Not Infected", "Infected"], yticklabels=["Not I
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
61/61 ━━━━━━━━━━━━━━━━━━━━ 99s 2s/step
```