OPENCV CODE FOR THE VIDEO DETECTION

```
import cv2
import numpy as np
from keras.models import load_model

# Load the Keras model and labels
model = load_model('converted_keras/keras_model.h5')
with open('converted_keras/labels.txt', 'r') as f:
    labels = f.read().splitlines()

# Initialize webcam
cap = cv2.VideoCapture(0)  # Use 0 for the default webcam, or provide the
appropriate index

while True:
    # Read frame from the webcam
    ret, frame = cap.read()

    if not ret:
        break

    # Preprocess the frame
    # Perform any necessary preprocessing on the frame before passing it to
the model

    # Resize the frame to match the input size of the model
    frame = cv2.resize(frame, (224, 224))

    # Convert the frame to the input format expected by the model (e.g., RGB,
normalized)
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame = frame.astype(np.float32) / 255.0

    # Perform prediction with the model
    # Reshape the frame to match the input shape expected by the model
    frame = np.expand_dims(frame, axis=0)

    # Make the prediction
    predictions = model.predict(frame)

    # Get the predicted class label
    class_index = np.argmax(predictions)
    class_label = labels[class_index]

    # Display the result
    frame = cv2.cvtColor(frame[0], cv2.COLOR_RGB2BGR)
    cv2.putText(frame, class_label, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 2)
    cv2.imshow('Live Detection', frame)

    # Check for 'q' key press to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture and close windows
```

```
cap.release()
cv2.destroyAllWindows()
```

OPENCV CODE FOR With IMAGE DETECTION

```
import cv2
import numpy as np
from keras.models import load_model

# Load the Keras model and labels
model = load_model('converted_keras/keras_model.h5')
with open('converted_keras/labels.txt', 'r') as f:
    labels = f.read().splitlines()

# Load the image
image_path = 'images/369.jpg'
image = cv2.imread(image_path)

# Preprocess the image
# Perform any necessary preprocessing on the image before passing it to the
model

# Resize the image to match the input size of the model
image = cv2.resize(image, (224, 224))

# Convert the image to the input format expected by the model (e.g., RGB,
normalized)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = image.astype(np.float32) / 255.0

# Perform prediction with the model
# Reshape the image to match the input shape expected by the model
image = np.expand_dims(image, axis=0)

# Make the prediction
predictions = model.predict(image)

# Get the predicted class label
class_index = np.argmax(predictions)
class_label = labels[class_index]

# Display the result
image = cv2.cvtColor(image[0], cv2.COLOR_RGB2BGR)
cv2.putText(image, class_label, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
255, 0), 2)
cv2.imshow('Image Detection', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```