```
import tensorflow as tf
from tensorflow import keras
import numpy as np
from PIL import Image

# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()

# Normalize the pixel values to be between 0 and 1
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

# Convert the labels to one-hot encoded vectors
y_train = keras.utils.to_categorical(y_train, num_classes=10)
y_test = keras.utils.to_categorical(y_test, num_classes=10)

# Define the model architecture
model = keras.models.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_data=(x_test, y_test))

# Save the trained model to a file
model.save("cifar10_model.h5")

# Load the saved model
model = keras.models.load_model("cifar10_model.h5")

# Load and preprocess the test image
img = Image.open("two.png")
img = img.resize((32, 32))
img_array = np.array(img)
img_array = img_array.astype("float32") / 255.0
img_array = np.expand_dims(img_array, axis=0)

# Make predictions on the test image
predictions = model.predict(img_array)

# Get the predicted class label
class_label = np.argmax(predictions)

# Print the predicted class label
print("Predicted class label:", class_label)
```

```
Epoch 1/10
782/782 [==============================] - 86s 109ms/step - loss: 1.6184 - accuracy: 0.4082 - val_loss: 1.3286 - val_accuracy: 0.5206
Epoch 2/10
782/782 [==============================] - 80s 102ms/step - loss: 1.2420 - accuracy: 0.5580 - val_loss: 1.1840 - val_accuracy: 0.5760
Epoch 3/10
782/782 [==============================] - 80s 102ms/step - loss: 1.0940 - accuracy: 0.6118 - val_loss: 1.0487 - val_accuracy: 0.6262
Epoch 4/10
782/782 [==============================] - 80s 102ms/step - loss: 0.9926 - accuracy: 0.6519 - val_loss: 0.9966 - val_accuracy: 0.6505
Epoch 5/10
782/782 [==============================] - 81s 103ms/step - loss: 0.9167 - accuracy: 0.6779 - val_loss: 0.9500 - val_accuracy: 0.6668
Epoch 6/10
782/782 [==============================] - 80s 102ms/step - loss: 0.8572 - accuracy: 0.7000 - val_loss: 0.9095 - val_accuracy: 0.6796
Epoch 7/10
782/782 [==============================] - 80s 103ms/step - loss: 0.8077 - accuracy: 0.7172 - val_loss: 0.9101 - val_accuracy: 0.6831
Epoch 8/10
782/782 [==============================] - 82s 104ms/step - loss: 0.7604 - accuracy: 0.7357 - val_loss: 0.8938 - val_accuracy: 0.6938
Epoch 9/10
782/782 [==============================] - 80s 102ms/step - loss: 0.7236 - accuracy: 0.7480 - val_loss: 0.8805 - val_accuracy: 0.6985
Epoch 10/10
782/782 [==============================] - 84s 107ms/step - loss: 0.6909 - accuracy: 0.7572 - val_loss: 0.9254 - val_accuracy: 0.6864
1/1 [==============================] - 0s 145ms/step
Predicted class label: 0
```