EXERCISE NO. 1
CNN FOR IMAGE CLASSIFICATION

AIM:
To write a program to demonstrate the working of CNN architecture to classify images.

ALGORITHM:

1. Import the necessary libraries.
2. Load the CIFAR-10 dataset.
3. Define the train-test split for training the CNN model
4. Build the CNN architecture
5. Compile and train the model.
6. Test the model for an unknown image.

PROGRAM:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt

cifar10_class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = models.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.4),
```

```
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_data=(x_test, y_test))
def classify_image(model, image_path):
    img = load_img(image_path, target_size=(32, 32))
    img_array = img_to_array(img)
    img_array = img_array / 255.0
    img_array = np.expand_dims(img_array, axis=0)
    predictions = model.predict(img_array)
    predicted_class_index = np.argmax(predictions)
    predicted_class_name = cifar10_class_names[predicted_class_index]
    import matplotlib.pyplot as plt
    plt.imshow(img)
    plt.title(f"Predicted: {predicted_class_name}", fontsize=16, fontweight="bold")
    plt.axis('off')
    plt.show()
    print(f"Predicted class: {predicted_class_name} (Confidence:
{predictions[0][predicted_class_index]:.2f})")

image_path = ".../frog-img.jpeg"
classify_image(model, image_path)
```

OUTPUT:

RESULT:

Thus the program has been successfully implemented and verified.