Neural network on cifar10 for image classification

In [33]:

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import mnist,cifar10,fashion_mnist
from tensorflow.keras import layers
from tensorflow.keras.layers import Embedding,Dense,SimpleRNN,LSTM,GRU,Dropout,Flatten
from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.applications import VGG19
from tensorflow.image import grayscale_to_rgb,resize
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import regularizers
from tensorflow.keras import optimizers
from tensorflow.keras.initializers import HeNormal,GlorotNormal
```

In [34]:

```
(train_images,train_labels),(test_images,test_labels)=cifar10.load_data()
train_images,test_images=train_images/255.0,test_images/255.0
train_labels,test_labels=to_categorical(train_labels),to_categorical(test_labels)
```

In [35]:

```
model=Sequential([
    Flatten(input_shape=(32,32,3)),
    Dense(256,activation='relu'),
    Dense(128,activation='relu'),
    Dense(64,activation='relu'),
    Dense(10,activation='softmax')
])
model.compile(metrics=['accuracy'],loss='categorical_crossentropy',optimizer='adam')
```

In [36]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_2 (Flatten)         (None, 3072)              0         
                                                                 
 dense_8 (Dense)             (None, 256)               786688    
                                                                 
 dense_9 (Dense)             (None, 128)               32896     
                                                                 
 dense_10 (Dense)            (None, 64)                8256      
                                                                 
 dense_11 (Dense)            (None, 10)                650       
                                                                 
=================================================================
Total params: 828490 (3.16 MB)
Trainable params: 828490 (3.16 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [37]:
```
history=model.fit(train_images,train_labels,epochs=5,validation_data=(test_images,test_labels))
loss,accuracy=model.evaluate(test_images,test_labels)
```

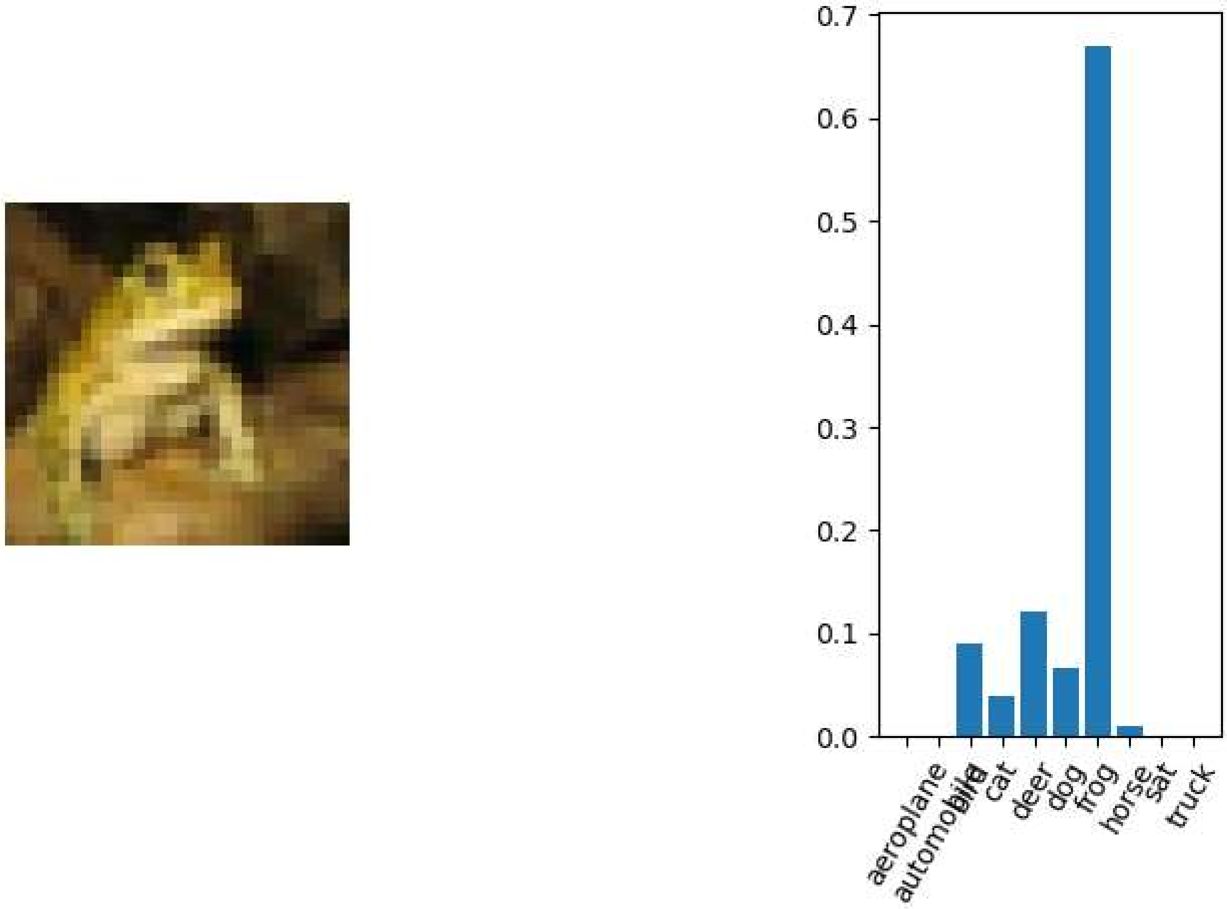
```
Epoch 1/5
1563/1563 [==============================] - 16s 9ms/step - loss: 1.8672 - accuracy: 0.3209 - val_loss: 1.73
81 - val_accuracy: 0.3764
Epoch 2/5
1563/1563 [==============================] - 14s 9ms/step - loss: 1.6832 - accuracy: 0.3936 - val_loss: 1.64
14 - val_accuracy: 0.4005
Epoch 3/5
1563/1563 [==============================] - 14s 9ms/step - loss: 1.6003 - accuracy: 0.4266 - val_loss: 1.65
71 - val_accuracy: 0.4034
Epoch 4/5
1563/1563 [==============================] - 14s 9ms/step - loss: 1.5546 - accuracy: 0.4427 - val_loss: 1.57
95 - val_accuracy: 0.4338
Epoch 5/5
1563/1563 [==============================] - 14s 9ms/step - loss: 1.5179 - accuracy: 0.4550 - val_loss: 1.52
94 - val_accuracy: 0.4508
313/313 [==============================] - 1s 3ms/step - loss: 1.5294 - accuracy: 0.4508
```

```
In [ ]:
```

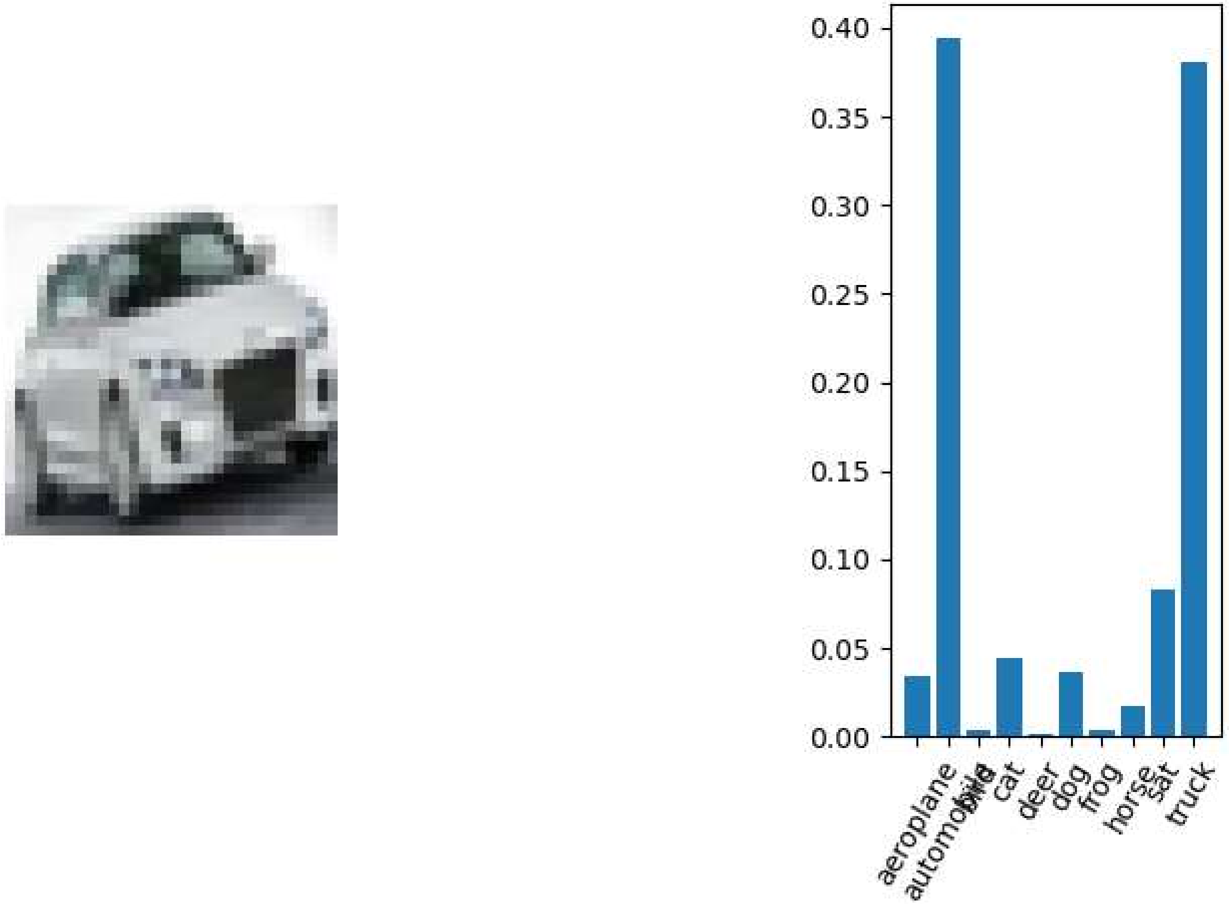
In [38]:

```
classes=['aeroplane','automobile','bird','cat','deer','dog','frog','horse','sat','truck']
for _ in range(3):
    index=np.random.randint(0,len(test_images))
    plt.subplot(1,3,1)
    plt.imshow(test_images[index])
    plt.axis('off')
    actual_label=test_labels[index]
    actual_label=np.argmax(actual_label)

    input_image=test_images[index]
    input_image=np.expand_dims(input_image,axis=0)
    predict=model.predict(input_image)
    probab=model.predict(input_image)[0]
    predicted_label=np.argmax(predict)
    print(f'actual_label : {classes[actual_label]} predicted_label: {classes[predicted_label]}')
    plt.subplot(1,3,3)
    plt.bar(classes,probab)
    plt.xticks(rotation=60)
    plt.tight_layout()

    plt.show()
```

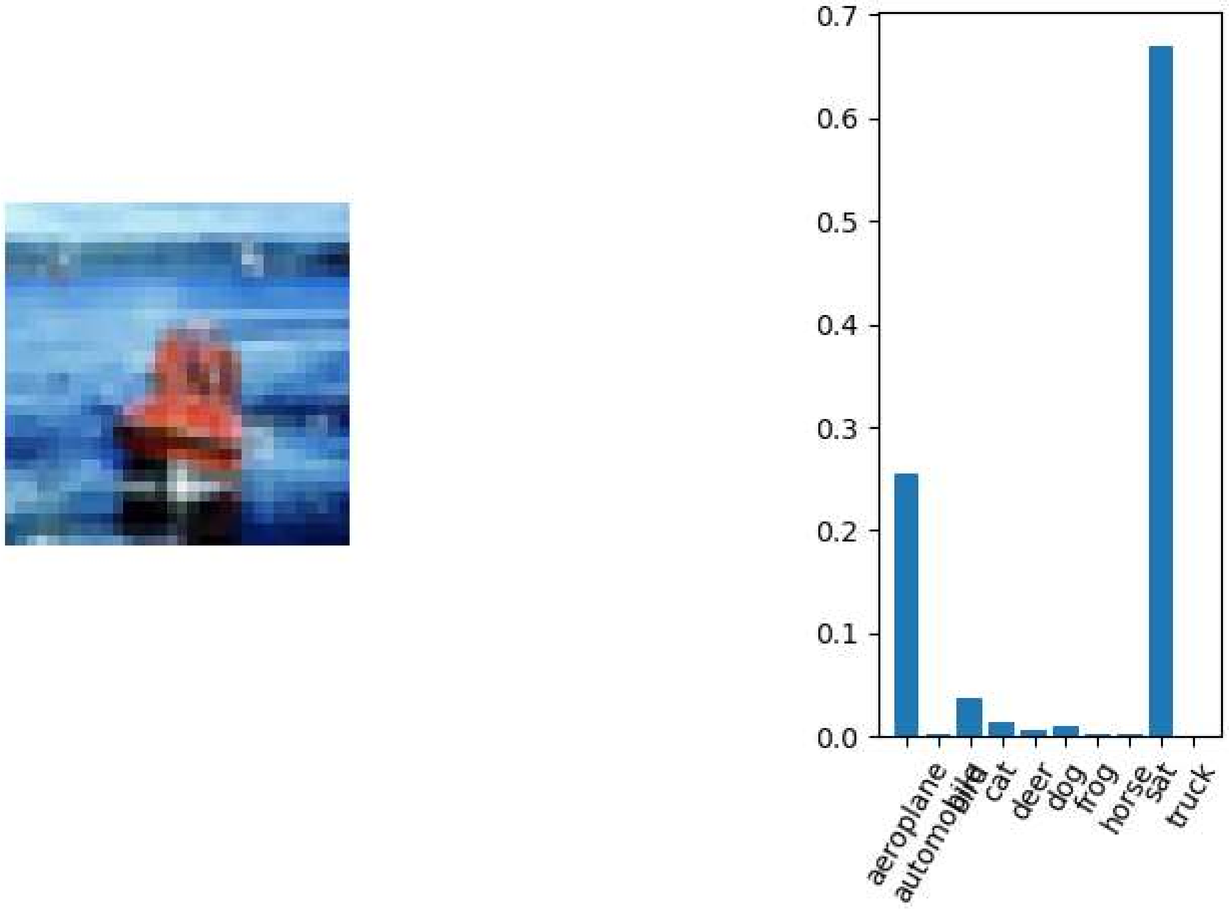
```
1/1 [==============================] - 0s 96ms/step
1/1 [==============================] - 0s 32ms/step
actual_label : frog predicted_label: frog
```



```
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
actual_label : automobile predicted_label: automobile
```



```
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
actual_label : sat predicted_label: sat
```


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
aeroplane
automobile
bird
cat
deer
dog
frog
horse
sat
truck

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```