In [ ]:
```
#Import libraries
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#Download the CIFAR10 dataset
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
```

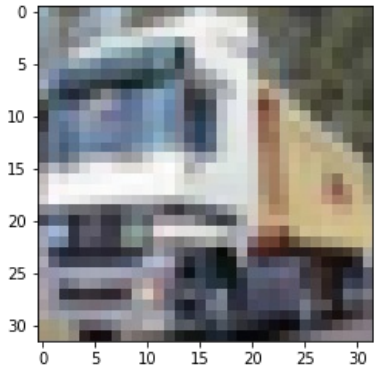
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 22s 0us/step
170508288/170498071 [==============================] - 22s 0us/step
```

In [3]:
```
#Normalize the pixels
train_images = train_images / 255.0
test_images = test_images / 255.0
```

In [4]:
```
#Get the class names of the labels
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

In [5]:
```
index = 1
plt.imshow(train_images[index])
class_names[int(train_labels[index])]
```

Out[5]:
```
'truck'
```



In [6]:
```
#Create a model
model = Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, input_dim=4, activation='relu'))
model.add(layers.Dense(10))

model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['a
```

```
2022-04-23 18:23:35.129534: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 18:23:35.188761: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 18:23:35.189626: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 18:23:35.190305: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1900] Ignoring visible gpu device
(device: 0, name: Quadro K1000M, pci bus id: 0000:01:00.0, compute capability: 3.0) with Cuda compute capability
3.0. The minimum required Cuda capability is 3.5.
2022-04-23 18:23:35.191103: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimi
zed with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critica
l operations:  SSE4.1 SSE4.2 AVX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

In [7]:
```
#Train the model
```

```
model.fit(train_images, train_labels, epochs=10, validation_data=(test_images, test_labels))
```

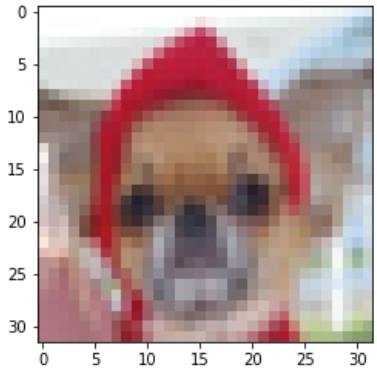
```
Epoch 1/10
1563/1563 [==============================] - 50s 32ms/step - loss: 1.5327 - accuracy: 0.4400 - val_loss: 1.2506 -
val_accuracy: 0.5569
Epoch 2/10
1563/1563 [==============================] - 62s 40ms/step - loss: 1.1630 - accuracy: 0.5896 - val_loss: 1.0881 -
val_accuracy: 0.6172
Epoch 3/10
1563/1563 [==============================] - 65s 42ms/step - loss: 1.0013 - accuracy: 0.6468 - val_loss: 0.9613 -
val_accuracy: 0.6585
Epoch 4/10
1563/1563 [==============================] - 66s 42ms/step - loss: 0.9084 - accuracy: 0.6833 - val_loss: 0.9382 -
val_accuracy: 0.6719
Epoch 5/10
1563/1563 [==============================] - 66s 42ms/step - loss: 0.8339 - accuracy: 0.7046 - val_loss: 0.9009 -
val_accuracy: 0.6868
Epoch 6/10
1563/1563 [==============================] - 67s 43ms/step - loss: 0.7782 - accuracy: 0.7282 - val_loss: 0.8923 -
val_accuracy: 0.6892
Epoch 7/10
1563/1563 [==============================] - 67s 43ms/step - loss: 0.7295 - accuracy: 0.7427 - val_loss: 0.9152 -
val_accuracy: 0.6957
Epoch 8/10
1563/1563 [==============================] - 67s 43ms/step - loss: 0.6853 - accuracy: 0.7607 - val_loss: 0.8713 -
val_accuracy: 0.7030
Epoch 9/10
1563/1563 [==============================] - 70s 45ms/step - loss: 0.6481 - accuracy: 0.7718 - val_loss: 0.8683 -
val_accuracy: 0.7102
Epoch 10/10
1563/1563 [==============================] - 68s 43ms/step - loss: 0.6152 - accuracy: 0.7833 - val_loss: 0.8907 -
val_accuracy: 0.7092
```

Out[7]:
```
<keras.callbacks.History at 0x7fd1644b6910>
```

In [8]:
```
#Test the model
y_pred = model.predict(test_images)
```

In [9]:
```
index = 168
plt.imshow(test_images[index])
class_names[y_pred[index].argmax()]
```

Out[9]:
```
'truck'
```



In [10]:
```
model.evaluate(test_images, test_labels, verbose=0)
```

Out[10]:
```
[0.8907056450843811, 0.7092000246047974]
```