# Small Multi - Image Classification Using Convolutional Neural Network (CNN)

## Importing Packages

In [2]:
```
import tensorflow as tf
from tensorflow.keras import models, layers, datasets
import numpy as np
import matplotlib.pyplot as plt
import random
```

## Load the Dataset

In [9]:
```
(X_train, y_train),(X_test,y_test) = datasets.cifar10.load_data()
X_train.shape
```

Out[9]: (50000, 32, 32, 3)

In [10]:
```
X_test.shape
```

Out[10]: (10000, 32, 32, 3)

In [11]:
```
y_train = y_train.reshape(-1,)
y_train[:5]
```

Out[11]: array([6, 9, 9, 4, 1], dtype=uint8)

In [12]:
```
y_test = y_test.reshape(-1,)
```

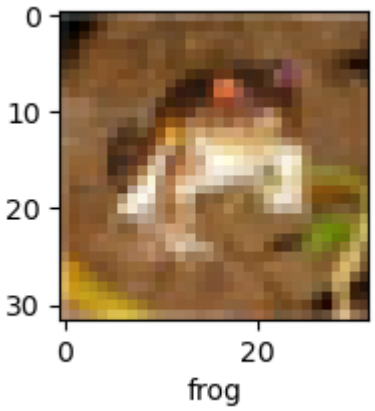
In [13]:
```
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

## Plotting Images

In [14]:
```
def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

In [15]:
```
# idx = random.randint(0, len(classes) - 1)
# plt.imshow(X_train[idx, :] * 255, cmap="gray")
# plt.figure(figsize = (15,2))
# plt.xlabel(classes[idx ])
# plt.show
```

In [16]:
```
plot_sample(X_train, y_train, 0)
```




## Normalizing

In [17]:
```
X_train = X_train / 255
X_test = X_test / 255
```

# CNN Model

In [18]:
```
cnn = models.Sequential([
# 1. first layer
            layers.Conv2D(filters=32, kernel_size = (3,3), activation='relu',input_shape=(32,32,3)),
            layers.MaxPooling2D((2,2)),

            layers.Conv2D(filters=64, kernel_size = (3,3), activation='relu'),
            layers.MaxPooling2D((2,2)),
#             layers.Dropout(0.4),



            layers.Conv2D(filters=128, kernel_size = (3,3), activation='relu'),
            layers.BatchNormalization(scale=False, center=True),
            layers.MaxPooling2D((2,2)),




# dense
            layers.Flatten(),
            layers.Dense(64, activation = "relu"),
            layers.Dense(10, activation = "softmax")
])
```

```
/Users/rsn/anaconda3/lib/python3.11/site-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [43]:
```
cnn.compile(loss = 'sparse_categorical_crossentropy', optimizer = 'adam', metrics = ["accuracy"])
```

In [ ]:
```
cnn.fit(X_train, y_train, epochs = 20)
```

```
Epoch 1/20
730/1563 ━━━━━━━━━━━━━━━━━━━━ 8s 10ms/step - accuracy: 0.3550 - loss: 1.7753
```

In [15]:
```
cnn.evaluate(X_test,y_test,1)
```

```
10000/10000 ━━━━━━━━━━━━━━━━━━━━ 5s 466us/step - accuracy: 0.7213 - loss: 1.3728
```

Out[15]:
```
[1.4209721088409424, 0.7150999903678894]
```

# Prediction of Model and Original Image

In [21]:
```
y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```

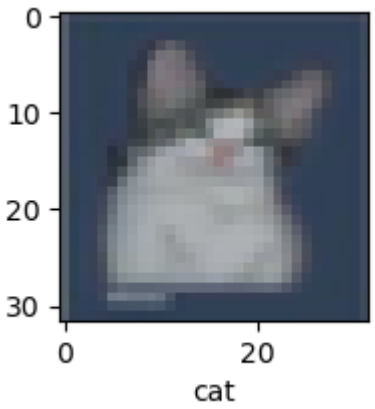
Out[21]:
```
array([[0.09844893, 0.10485107, 0.09731167, 0.11000157, 0.09579714,
        0.09356013, 0.09159243, 0.10862612, 0.09581267, 0.1039983 ],
       [0.09756054, 0.10869686, 0.09831304, 0.11347739, 0.09365536,
        0.08900777, 0.08908912, 0.11145382, 0.09374046, 0.10500567],
       [0.09712002, 0.10856374, 0.09847423, 0.11379205, 0.0924219 ,
        0.08906619, 0.09198888, 0.11042493, 0.09461053, 0.10353757],
       [0.09629702, 0.10787153, 0.09891844, 0.11201771, 0.09233502,
        0.09007872, 0.09003609, 0.10949775, 0.09598594, 0.10696188],
       [0.09659911, 0.10564169, 0.09859926, 0.10994371, 0.09481632,
        0.09103122, 0.09218573, 0.10888603, 0.09557738, 0.10671964]],
      dtype=float32)
```

In [22]:
```
# y_classes = [np.argmax(element) for element in y_pred]
# or
y_classes = np.argmax(y_pred, axis=1)

y_classes[:5]
```

Out[22]:
```
array([3, 3, 3, 3, 3])
```

In [33]:
```
a = random.randint(0,9999)
plot_sample(X_test, y_test,a)

print("Model says its a :",classes[y_classes[a]])
```

```
Model says its a : cat
```




In [ ]: