```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
```

```
(X_train, y_train), (X_test,y_test) = datasets.cifar10.load_data()
X_train.shape
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-
python.tar.gz
170498071/170498071 [==============================] - 3s 0us/step
```

```
(50000, 32, 32, 3)
```

```
X_test.shape
```

```
(10000, 32, 32, 3)
```

Here we see there are 50000 training images and 1000 test images

```
y_train.shape
```

```
(50000, 1)
```

```
y_train[:5]
```

```
array([[6],
       [9],
       [9],
       [4],
       [1]], dtype=uint8)
```

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

```
array([6, 9, 9, 4, 1], dtype=uint8)
```

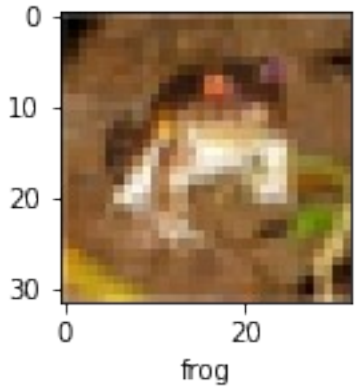
```
y_test = y_test.reshape(-1,)
```

```
classes =
["airplane","automobile","bird","cat","deer","dog","frog","horse","shi
p","truck"]
```

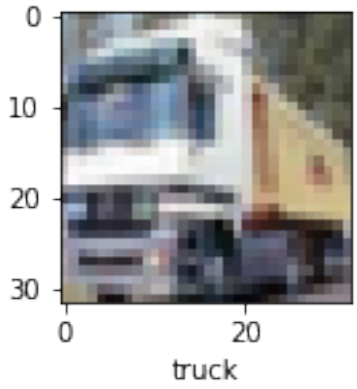
Let's plot some images to see what they are

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

plot_sample(X_train, y_train, 0)
```




```
plot_sample(X_train, y_train, 1)
```




```
X_train = X_train / 255.0
X_test = X_test / 255.0

cnn = models.Sequential([
    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.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

cnn.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 [==============================] - 79s 49ms/step - loss:
1.4304 - accuracy: 0.4877
Epoch 2/10
1563/1563 [==============================] - 76s 48ms/step - loss:
1.0872 - accuracy: 0.6208
Epoch 3/10
1563/1563 [==============================] - 75s 48ms/step - loss:
0.9625 - accuracy: 0.6629
Epoch 4/10
1563/1563 [==============================] - 76s 48ms/step - loss:
0.8746 - accuracy: 0.6967
Epoch 5/10
1563/1563 [==============================] - 76s 48ms/step - loss:
0.8097 - accuracy: 0.7177
Epoch 6/10
1563/1563 [==============================] - 74s 48ms/step - loss:
0.7520 - accuracy: 0.7385
Epoch 7/10
1563/1563 [==============================] - 77s 49ms/step - loss:
0.6968 - accuracy: 0.7585
Epoch 8/10
1563/1563 [==============================] - 75s 48ms/step - loss:
0.6520 - accuracy: 0.7721
Epoch 9/10
1563/1563 [==============================] - 76s 48ms/step - loss:
0.6066 - accuracy: 0.7897
Epoch 10/10
1563/1563 [==============================] - 76s 48ms/step - loss:
0.5688 - accuracy: 0.8015

<keras.callbacks.History at 0x7faeb10eb820>

cnn.evaluate(X_test,y_test)

313/313 [==============================] - 5s 16ms/step - loss: 0.9197
- accuracy: 0.6978

[0.9196503758430481, 0.6977999806404114]

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

313/313 [==============================] - 4s 14ms/step

array([[4.85083181e-03, 1.26784071e-04, 6.86467800e-04, 5.41134715e-
01,
        8.10673926e-04, 4.48421389e-01, 3.40047991e-03, 8.01703063e-
05,
        3.85021616e-04, 1.03420614e-04],
       [1.67427193e-02, 3.46563905e-01, 2.62802132e-05, 1.63778568e-
05,
```

```
        1.90715218e-06, 2.17040252e-07, 6.20016726e-05, 2.02985959e-
07,
        6.15407109e-01, 2.11793482e-02],
       [1.84481621e-01, 5.05710125e-01, 2.15471047e-03, 8.69720336e-
03,
        8.31601094e-04, 7.76328205e-04, 1.03307259e-03, 9.29076225e-
04,
        2.23247096e-01, 7.21391812e-02],
       [9.53086972e-01, 8.51027109e-03, 5.53338882e-03, 1.38944772e-
03,
        2.09355028e-03, 2.46619802e-05, 1.98827711e-05, 6.43114690e-
05,
        2.91973520e-02, 8.02646173e-05],
       [3.65029587e-06, 3.47015293e-06, 4.03198740e-03, 8.06742907e-
03,
        5.54284930e-01, 3.55270458e-04, 4.33179915e-01, 4.16099147e-
06,
        6.89612134e-05, 1.35822077e-07]], dtype=float32)
```

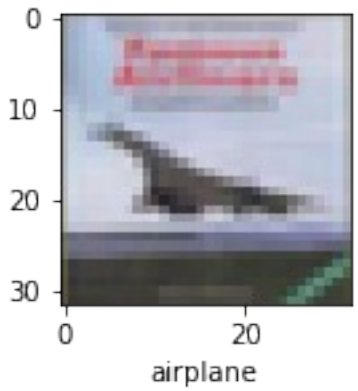
```
y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

```
[3, 8, 1, 0, 4]
```

```
y_test[:5]
```

```
array([3, 8, 8, 0, 6], dtype=uint8)
```

```
plot_sample(X_test, y_test,3)
```




```
classes[y_classes[3]]
```

```
{"type":"string"}
```

```
classes[y_classes[3]]
```

```
{"type":"string"}
```