```
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 [==============================] - 2s 0us/step
(50000, 32, 32, 3)
```

```
X_test.shape
```

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

```
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]
```

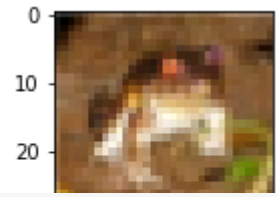
```
array([6, 9, 9, 4, 1], dtype=uint8)
```

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

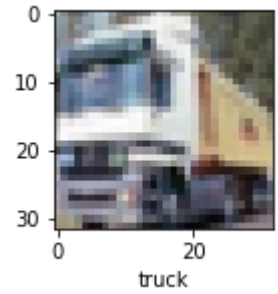
```
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"
```

```
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
```

```
ann = models.Sequential([
 layers.Flatten(input_shape=(32,32,3)),
 layers.Dense(3000, activation='relu'),
 layers.Dense(1000, activation='relu'),
 layers.Dense(10, activation='softmax')
 ])
ann.compile(optimizer='SGD',
 loss='sparse_categorical_crossentropy',
 metrics=['accuracy'])
ann.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5
1563/1563 [==============================] - 117s 74ms/step - loss: 1.8136 - accuracy
Epoch 2/5
1563/1563 [==============================] - 113s 73ms/step - loss: 1.6230 - accuracy
Epoch 3/5
1563/1563 [==============================] - 117s 75ms/step - loss: 1.5404 - accuracy
Epoch 4/5
1563/1563 [==============================] - 115s 74ms/step - loss: 1.4822 - accuracy
Epoch 5/5
1563/1563 [==============================] - 112s 72ms/step - loss: 1.4307 - accuracy
<keras.callbacks.History at 0x7f38c14279d0>
```

```
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 [==============================] - 74s 47ms/step - loss: 1.4737 - accuracy
Epoch 2/10
1563/1563 [==============================] - 73s 47ms/step - loss: 1.1173 - accuracy
Epoch 3/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.9792 - accuracy
Epoch 4/10
 746/1563 [=============>................] - ETA: 37s - loss: 0.8995 - accuracy: 0.68
```

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

```
Epoch 1/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.5631 - accuracy
Epoch 2/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.5254 - accuracy
Epoch 3/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.4986 - accuracy
Epoch 4/10
1563/1563 [==============================] - 74s 47ms/step - loss: 0.4654 - accuracy
Epoch 5/10
1563/1563 [==============================] - 71s 46ms/step - loss: 0.4351 - accuracy
Epoch 6/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.4030 - accuracy
Epoch 7/10
1563/1563 [==============================] - 71s 45ms/step - loss: 0.3814 - accuracy
Epoch 8/10
1563/1563 [==============================] - 72s 46ms/step - loss: 0.3557 - accuracy
Epoch 9/10
1563/1563 [==============================] - 71s 46ms/step - loss: 0.3278 - accuracy
Epoch 10/10
1563/1563 [==============================] - 71s 45ms/step - loss: 0.3093 - accuracy
<keras.callbacks.History at 0x7f38c571a090>
```

```
cnn.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 5s 15ms/step - loss: 1.3576 - accuracy: 0
[1.357582449913025, 0.6814000010490417]
```

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

```
313/313 [==============================] - 5s 14ms/step
array([[7.84178695e-08, 1.16472471e-07, 1.10133578e-06, 9.91877019e-01,
        1.60739845e-07, 8.08447786e-03, 3.42374478e-05, 2.62294378e-07,
        1.51889412e-06, 9.91471211e-07],
       [2.92035820e-05, 8.56658936e-01, 6.30652586e-10, 1.40098035e-12,
        1.34018136e-13, 3.83953909e-13, 8.43242078e-17, 1.42271399e-16,
        1.43310830e-01, 1.06782534e-06],
       [9.81403631e-04, 9.64489937e-01, 1.07726904e-04, 2.57934700e-03,
        3.48145863e-06, 1.22636793e-05, 2.17049617e-07, 2.12477735e-05,
        2.31610052e-02, 8.64319131e-03],
       [9.09422040e-01, 1.15823420e-03, 7.03273788e-02, 8.80572770e-06,
        9.95027833e-04, 4.15378487e-09, 7.60254579e-06, 2.57810595e-09,
        1.79500952e-02, 1.30749599e-04],
       [8.30017695e-13, 1.67859966e-07, 2.28815734e-01, 5.55181086e-01,
        1.52537059e-02, 8.19338857e-06, 2.00741127e-01, 5.36732159e-10,
        1.06408207e-08, 8.08646227e-14]], dtype=float32)
```

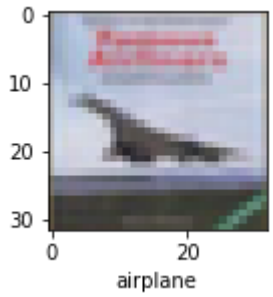
```
y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

```
[3, 1, 1, 0, 3]
```

```
y_test[:5]
```

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

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




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

```
'airplane'
```

Colab paid products - Cancel contracts here

✓ 0s completed at 10:10 PM