```
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)
```

there are 50k samples, 32x32 image is each sample, 3 is for RGB channels

```
X_test.shape
```

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

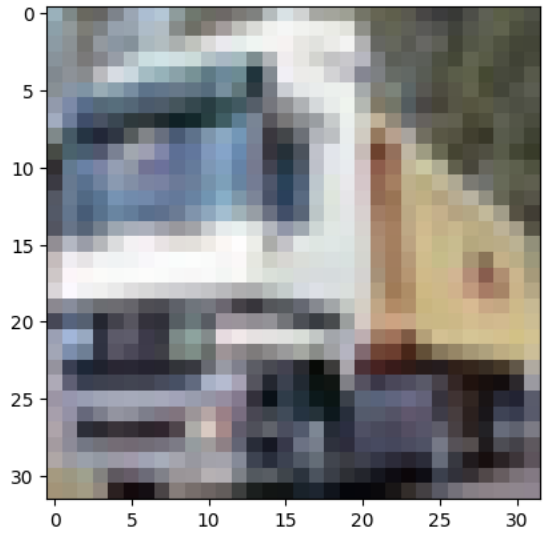
```
X_train[0]
```

```
array([[[ 59,  62,  63],
        [ 43,  46,  45],
        [ 50,  48,  43],
        ...,
        [158, 132, 108],
        [152, 125, 102],
        [148, 124, 103]],

       [[ 16,  20,  20],
        [  0,   0,   0],
        [ 18,   8,   0],
        ...,
        [123,  88,  55],
        [119,  83,  50],
        [122,  87,  57]],

       [[ 25,  24,  21],
        [ 16,   7,   0],
        [ 49,  27,   8],
        ...,
        [118,  84,  50],
        [120,  84,  50],
        [109,  73,  42]],

       ...,

       [[208, 170,  96],
        [201, 153,  34],
        [198, 161,  26],
        ...,
        [160, 133,  70],
        [ 56,  31,   7],
        [ 53,  34,  20]],

       [[180, 139,  96],
        [173, 123,  42],
        [186, 144,  30],
        ...,
        [184, 148,  94],
        [ 97,  62,  34],
        [ 83,  53,  34]],

       [[177, 144, 116],
        [168, 129,  94],
        [179, 142,  87],
        ...,
        [216, 184, 140],
        [151, 118,  84],
        [123,  92,  72]]], dtype=uint8)
```

32x32 array , into 3 RGB channels

```
# matplotlib is imported as plt
plt.imshow(X_train[1])
```

```
<matplotlib.image.AxesImage at 0x7f788c123580>
```



```
y_train[:5]
```

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

```
# reshaping for convenience
y_train = y_train.reshape(-1,)
y_train[:5]
```

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

so earlier we had a 2D array; now we deal with a 1D array.

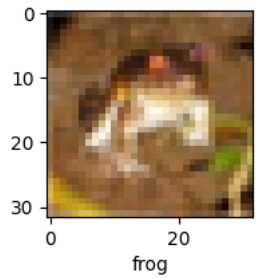
```
classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
```

```
classes[9]
```

```
'truck'
```

```
#controlling the size of the image
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)
```



if you notice now then the label is getting printed as well, ie, "frog"

```
# normalizing the values
X_train = X_train/255
X_test = X_test / 255
```

```
ann = models.Sequential([
        layers.Flatten(input_shape = (32,32,3)),
        layers.Dense(3000, activation='relu'),
        layers.Dense(1000, activation='relu'),
        layers.Dense(10, activation='sigmoid')
])

ann.compile(optimizer='SGD',
            loss= 'sparse_categorical_crossentropy',
            metrics=['accuracy'])

ann.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5
1563/1563 [==============================] - 142s 90ms/step - loss: 1.8131 - accuracy: 0.3532
Epoch 2/5
1563/1563 [==============================] - 140s 89ms/step - loss: 1.6197 - accuracy: 0.4306
Epoch 3/5
1563/1563 [==============================] - 144s 92ms/step - loss: 1.5384 - accuracy: 0.4586
Epoch 4/5
1563/1563 [==============================] - 142s 91ms/step - loss: 1.4778 - accuracy: 0.4807
Epoch 5/5
1563/1563 [==============================] - 148s 95ms/step - loss: 1.4286 - accuracy: 0.4989
<keras.callbacks.History at 0x7f7880d0f7c0>
```

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

```
313/313 [==============================] - 10s 31ms/step - loss: 1.5269 - accuracy: 0.4508
[1.5269345045089722, 0.45080000162124634]
```

```
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

```
313/313 [==============================] - 11s 35ms/step
Classification Report:
                precision    recall  f1-score   support

           0       0.57      0.44      0.49      1000
           1       0.61      0.60      0.61      1000
           2       0.39      0.23      0.29      1000
           3       0.39      0.21      0.27      1000
           4       0.43      0.37      0.39      1000
           5       0.56      0.21      0.31      1000
           6       0.32      0.81      0.46      1000
           7       0.80      0.26      0.40      1000
           8       0.43      0.80      0.56      1000
           9       0.50      0.57      0.54      1000

    accuracy                           0.45     10000
   macro avg       0.50      0.45      0.43     10000
weighted avg       0.50      0.45      0.43     10000
```

```
cnn = models.Sequential([

        #cnn
        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)),

        #dense
        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 [==============================] - 78s 49ms/step - loss: 1.4663 - accuracy: 0.4717
Epoch 2/10
1563/1563 [==============================] - 75s 48ms/step - loss: 1.1297 - accuracy: 0.6021
Epoch 3/10
1563/1563 [==============================] - 75s 48ms/step - loss: 0.9956 - accuracy: 0.6529
Epoch 4/10
1563/1563 [==============================] - 74s 47ms/step - loss: 0.9096 - accuracy: 0.6819
Epoch 5/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.8442 - accuracy: 0.7064
Epoch 6/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.7888 - accuracy: 0.7250
Epoch 7/10
1563/1563 [==============================] - 73s 46ms/step - loss: 0.7443 - accuracy: 0.7398
Epoch 8/10
1563/1563 [==============================] - 73s 47ms/step - loss: 0.7013 - accuracy: 0.7552
Epoch 9/10
1563/1563 [==============================] - 74s 47ms/step - loss: 0.6622 - accuracy: 0.7675
Epoch 10/10
1563/1563 [==============================] - 78s 50ms/step - loss: 0.6269 - accuracy: 0.7811
<keras.callbacks.History at 0x7f780197f130>
```

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

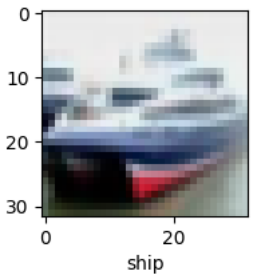
```
313/313 [==============================] - 8s 25ms/step - loss: 0.9152 - accuracy: 0.6953
[0.9151604771614075, 0.6952999830245972]
```

convolutional network just built!

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

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

```
plot_sample(X_test, y_test, 1)
```




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

```
313/313 [==============================] - 6s 17ms/step
array([[1.6678743e-03, 7.2006072e-04, 1.6014285e-02, 7.1116877e-01,
        5.9671287e-04, 1.7460862e-02, 2.4232103e-01, 1.3799129e-05,
        6.6816369e-03, 3.3548698e-03],
       [1.8638398e-03, 3.1010022e-03, 1.4859506e-07, 1.0354674e-08,
        8.2967917e-09, 1.3471672e-08, 9.9533104e-11, 2.6328841e-09,
        9.9503243e-01, 2.3640209e-06],
       [8.1125259e-02, 2.3380245e-01, 2.3880699e-03, 1.4838714e-03,
        1.7417060e-03, 1.1598059e-03, 1.6730465e-04, 5.5049188e-03,
        6.0437799e-01, 6.8248644e-02],
       [9.3866336e-01, 1.0425759e-03, 4.2153695e-03, 2.9870498e-05,
        9.7421452e-04, 2.2778247e-06, 1.7789321e-06, 9.9856125e-06,
        5.5057354e-02, 3.1296415e-06],
       [1.9959541e-06, 6.3375446e-05, 5.2178912e-02, 6.9167847e-03,
        8.5134238e-01, 2.8817268e-04, 8.9186780e-02, 3.5640819e-06,
        1.6082502e-05, 1.9300799e-06]], dtype=float32)
```

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

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

```
y_test[:5]
```

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

```
classes
```

```
['airplane',
 'automobile',
 'bird',
 'cat',
 'deer',
 'dog',
 'frog',
 'horse',
 'ship',
 'truck']
```

```
print("Classification Report : \n", classification_report(y_test, y_classes))
```

```
Classification Report :
               precision    recall  f1-score   support

           0       0.71      0.71      0.71      1000
           1       0.88      0.75      0.81      1000
           2       0.53      0.67      0.59      1000
           3       0.53      0.50      0.52      1000
           4       0.67      0.64      0.65      1000
           5       0.65      0.58      0.61      1000
           6       0.78      0.75      0.77      1000
           7       0.81      0.71      0.76      1000
           8       0.70      0.88      0.78      1000
           9       0.76      0.77      0.77      1000

    accuracy                           0.70     10000
   macro avg       0.70      0.70      0.70     10000
weighted avg       0.70      0.70      0.70     10000
```

✓ 0s completed at 1:13 PM