CIFAR-10 with CNN

In [ ]:

```
import numpy as np
import pandas as pd

from keras.datasets import cifar10
from keras import layers, models

import matplotlib.pyplot as plt
```

Dataset

In [ ]:

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

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.ta
r.gz
170500096/170498071 [==============================] - 11s 0us/step
170508288/170498071 [==============================] - 11s 0us/step
```

Out[ ]:

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

Reshaping the dataset

In [ ]:

```
y_train[:5] #it is a 2d array
```

Out[ ]:

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

Converting 2d array to 1d

In [ ]:

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

Out[ ]:

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

In [ ]:

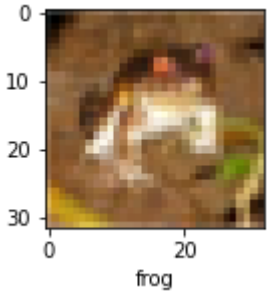
```
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","tru
ck"]
```

In [ ]:

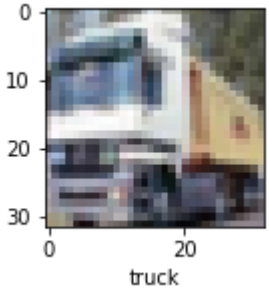
```
index=0
def plt_image(X,y,index):
    for index in range(10):
        plt.figure(figsize=(15,2))
        plt.imshow(X[index])
        plt.xlabel(classes[y[index]])
```

In [ ]:

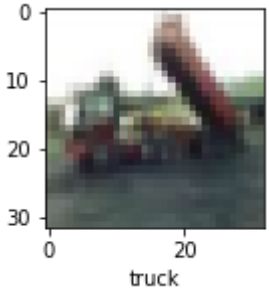
```
plt_image(X_train,y_train,0)
```



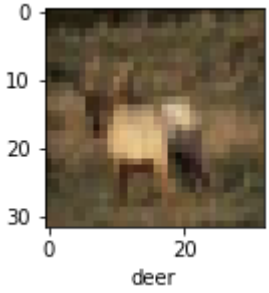
frog



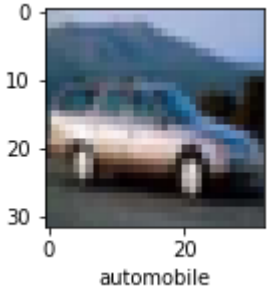
truck



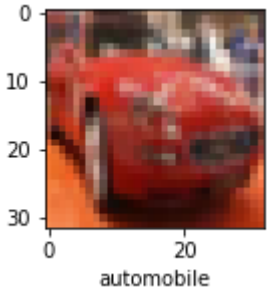
truck



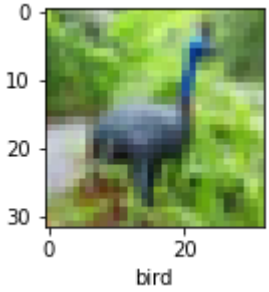
deer



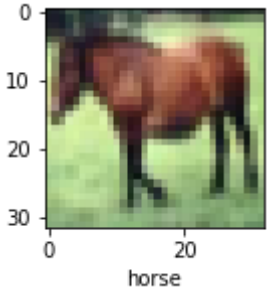
automobile



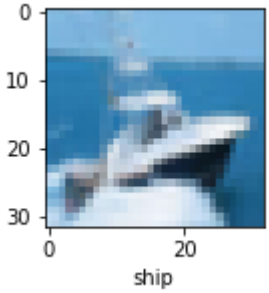
automobile



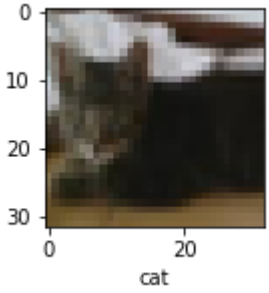
bird



horse



ship



cat

In [ ]:

```
X_train[0]
```

Out[ ]:

```
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)
```

The values are between 0 to 255 here.

Normalization

In [ ]:

```
X_train = X_train / 255
X_test  = X_test / 255
```

In [ ]:

```
X_test[0]
```

Out[ ]:

```
array([[[0.61960784, 0.43921569, 0.19215686],
        [0.62352941, 0.43529412, 0.18431373],
        [0.64705882, 0.45490196, 0.2       ],
        ...,
        [0.5372549 , 0.37254902, 0.14117647],
        [0.49411765, 0.35686275, 0.14117647],
        [0.45490196, 0.33333333, 0.12941176]],

       [[0.59607843, 0.43921569, 0.2       ],
        [0.59215686, 0.43137255, 0.15686275],
        [0.62352941, 0.44705882, 0.17647059],
        ...,
        [0.53333333, 0.37254902, 0.12156863],
        [0.49019608, 0.35686275, 0.1254902 ],
        [0.46666667, 0.34509804, 0.13333333]],

       [[0.59215686, 0.43137255, 0.18431373],
        [0.59215686, 0.42745098, 0.12941176],
        [0.61960784, 0.43529412, 0.14117647],
        ...,
        [0.54509804, 0.38431373, 0.13333333],
        [0.50980392, 0.37254902, 0.13333333],
        [0.47058824, 0.34901961, 0.12941176]],

       ...,

       [[0.26666667, 0.48627451, 0.69411765],
        [0.16470588, 0.39215686, 0.58039216],
        [0.12156863, 0.34509804, 0.5372549 ],
        ...,
        [0.14901961, 0.38039216, 0.57254902],
        [0.05098039, 0.25098039, 0.42352941],
        [0.15686275, 0.33333333, 0.49803922]],

       [[0.23921569, 0.45490196, 0.65882353],
        [0.19215686, 0.4       , 0.58039216],
        [0.1372549 , 0.33333333, 0.51764706],
        ...,
        [0.10196078, 0.32156863, 0.50980392],
        [0.11372549, 0.32156863, 0.49411765],
        [0.07843137, 0.25098039, 0.41960784]],

       [[0.21176471, 0.41960784, 0.62745098],
        [0.21960784, 0.41176471, 0.58431373],
        [0.17647059, 0.34901961, 0.51764706],
        ...,
        [0.09411765, 0.30196078, 0.48627451],
        [0.13333333, 0.32941176, 0.50588235],
        [0.08235294, 0.2627451 , 0.43137255]]])
```

ANN for classification

In [ ]:

```
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 [==============================] - 127s 81ms/step - loss: 1.8143
- accuracy: 0.3543
Epoch 2/5
1563/1563 [==============================] - 125s 80ms/step - loss: 1.6223
- accuracy: 0.4268
Epoch 3/5
1563/1563 [==============================] - 127s 81ms/step - loss: 1.5372
- accuracy: 0.4572
Epoch 4/5
1563/1563 [==============================] - 126s 81ms/step - loss: 1.4779
- accuracy: 0.4776
Epoch 5/5
1563/1563 [==============================] - 126s 81ms/step - loss: 1.4302
- accuracy: 0.4958
```

Out[ ]:

```
<keras.callbacks.History at 0x7f6add94a4d0>
```

In [ ]:

```

```

Classification Report

In [ ]:

```
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))
```

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

           0       0.47      0.64      0.54      1000
           1       0.64      0.58      0.61      1000
           2       0.47      0.25      0.32      1000
           3       0.40      0.16      0.23      1000
           4       0.48      0.33      0.39      1000
           5       0.31      0.56      0.40      1000
           6       0.61      0.44      0.51      1000
           7       0.40      0.72      0.51      1000
           8       0.76      0.37      0.50      1000
           9       0.49      0.64      0.56      1000

    accuracy                           0.47     10000
   macro avg       0.50      0.47      0.46     10000
weighted avg       0.50      0.47      0.46     10000
```

CNN MODEL

In [ ]:

```
cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(32, 3
2, 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')
])
```

In [ ]:

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

In [ ]:

```
cnn.fit(X_train, y_train, epochs=20)
```

```
Epoch 1/20
1563/1563 [==============================] - 67s 43ms/step - loss: 1.4692
- accuracy: 0.4734
Epoch 2/20
1563/1563 [==============================] - 67s 43ms/step - loss: 1.1083
- accuracy: 0.6116
Epoch 3/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.9659
- accuracy: 0.6629
Epoch 4/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.8788
- accuracy: 0.6938
Epoch 5/20
1563/1563 [==============================] - 68s 44ms/step - loss: 0.8090
- accuracy: 0.7180
Epoch 6/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.7484
- accuracy: 0.7387
Epoch 7/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.6981
- accuracy: 0.7570
Epoch 8/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.6504
- accuracy: 0.7720
Epoch 9/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.6099
- accuracy: 0.7861
Epoch 10/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.5717
- accuracy: 0.7977
Epoch 11/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.5322
- accuracy: 0.8113
Epoch 12/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.4935
- accuracy: 0.8239
Epoch 13/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.4600
- accuracy: 0.8368
Epoch 14/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.4268
- accuracy: 0.8476
Epoch 15/20
1563/1563 [==============================] - 68s 44ms/step - loss: 0.3927
- accuracy: 0.8603
Epoch 16/20
1563/1563 [==============================] - 69s 44ms/step - loss: 0.3649
- accuracy: 0.8696
Epoch 17/20
1563/1563 [==============================] - 69s 44ms/step - loss: 0.3387
- accuracy: 0.8778
Epoch 18/20
1563/1563 [==============================] - 68s 44ms/step - loss: 0.3140
- accuracy: 0.8873
Epoch 19/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.2873
- accuracy: 0.8977
Epoch 20/20
1563/1563 [==============================] - 68s 43ms/step - loss: 0.2707
- accuracy: 0.9028
```

Out[ ]:

```
<keras.callbacks.History at 0x7f6ada121810>
```

Evaluation

In [ ]:

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

```
313/313 [==============================] - 5s 15ms/step - loss: 1.4760 - a
ccuracy: 0.6761
```

Out[ ]:

```
[1.4759913682937622, 0.6761000156402588]
```

In [ ]:

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

Out[ ]:

```
array([[2.1079543e-06, 1.2484081e-08, 3.1694351e-05, 9.8148823e-01,
        1.9713311e-06, 1.8334292e-02, 1.5837455e-05, 7.4838346e-05,
        5.1032985e-05, 6.1166494e-09],
       [2.1921962e-09, 1.8000806e-06, 6.3891671e-13, 1.1895669e-16,
        6.3159111e-17, 8.3291897e-19, 6.3771132e-17, 5.9852521e-20,
        9.9999821e-01, 3.1482678e-08],
       [3.0281221e-02, 3.0862447e-04, 2.3356504e-06, 1.1458596e-06,
        3.6438598e-07, 1.9242745e-07, 2.7501443e-07, 1.7891272e-06,
        9.6937388e-01, 3.0205640e-05],
       [9.9994063e-01, 5.4400434e-10, 1.7710379e-05, 2.6976037e-09,
        1.8528180e-09, 8.3206337e-14, 2.4559636e-15, 9.5902758e-11,
        4.1588148e-05, 3.2746703e-09],
       [3.9562481e-10, 3.6335422e-07, 4.6247700e-03, 7.8240000e-03,
        4.7680828e-01, 4.0961979e-05, 5.1070124e-01, 2.7086397e-11,
        4.3380399e-07, 7.4611340e-13]], dtype=float32)
```

In [ ]:

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

Out[ ]:

```
[3, 8, 8, 0, 6]
```

In [ ]:

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

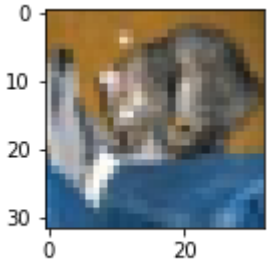
Out[ ]:

```
array([3, 8, 8, ..., 5, 1, 7], dtype=uint8)
```

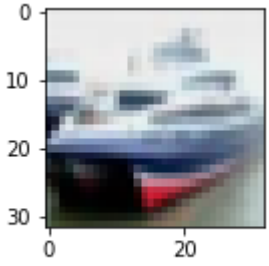
PLotting

In [ ]:

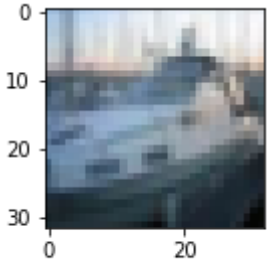
```
plt_image(X_test,y_classes,0)
```



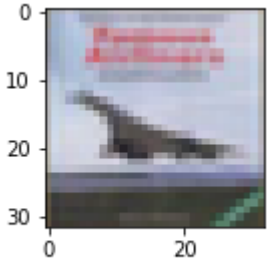
cat



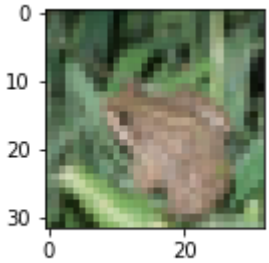
ship



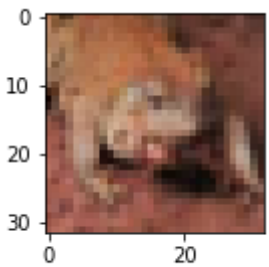
ship



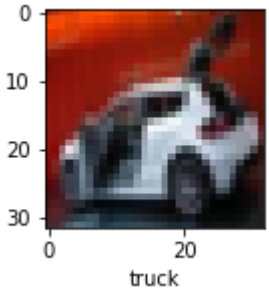
airplane

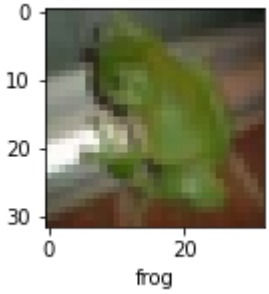


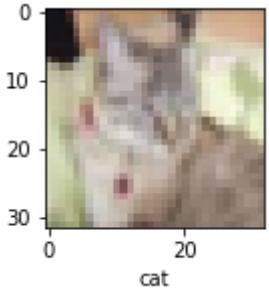
frog

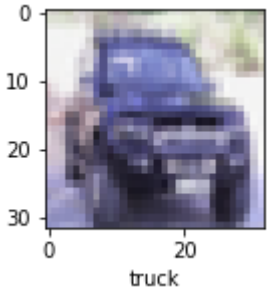


frog









Class

In [ ]:

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

Out[ ]:

```
'airplane'
```