Step1:Import packages

```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,  Flatten
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Step2:Load the Datasets

```
cifar10 = cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((50000, 32, 32, 3), (10000, 32, 32, 3), (50000, 1), (10000, 1))
```

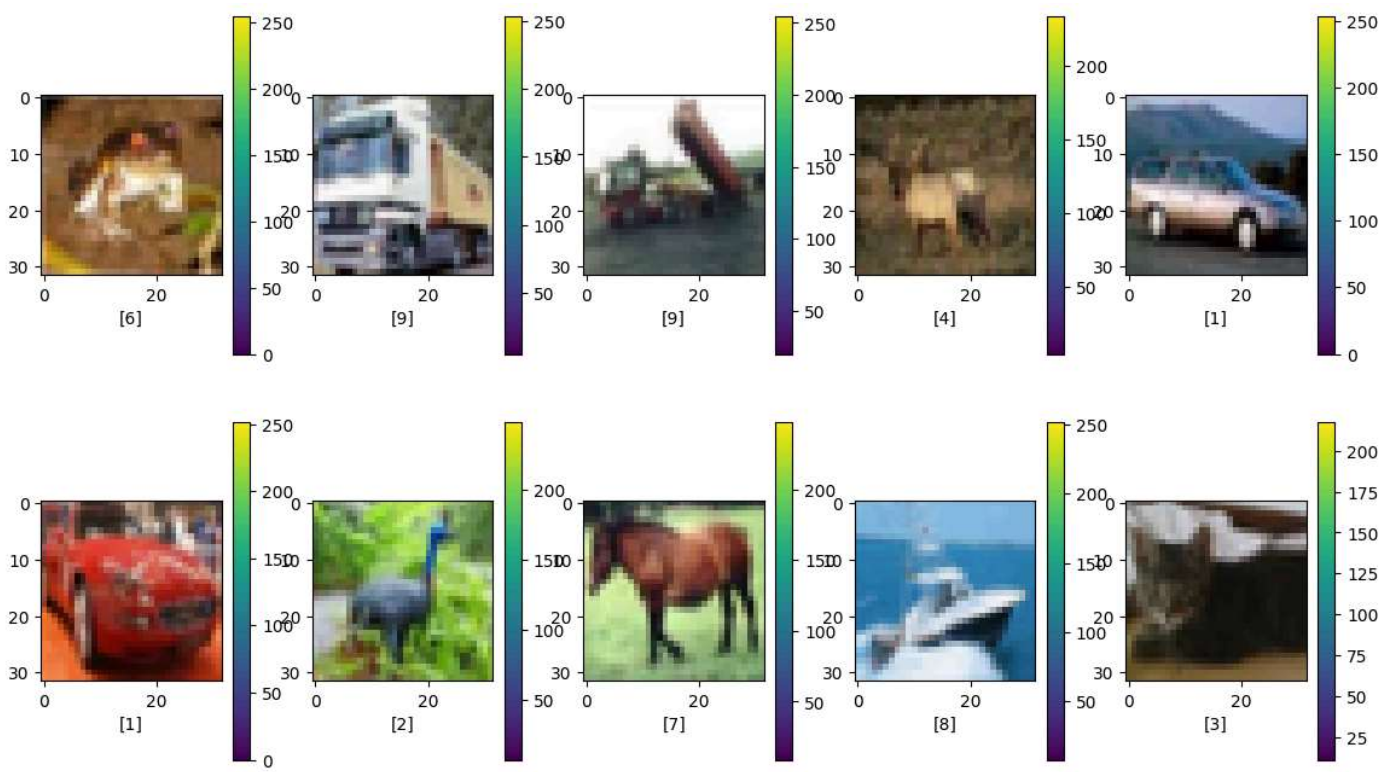
```
x_train[0]
```

```
ndarray (32, 32, 3) show data
```

```
min(y_train),max(y_train)
```

```
(array([0], dtype=uint8), array([9], dtype=uint8))
```

```
plt.figure(figsize=(14,8))
for i in range(10):
  plt.subplot(2,5,i+1)
  plt.imshow(x_train[i])
  plt.xlabel(y_train[i])
  #plt.title(class_names[y_train[i]])
  plt.colorbar()
plt.show()
```



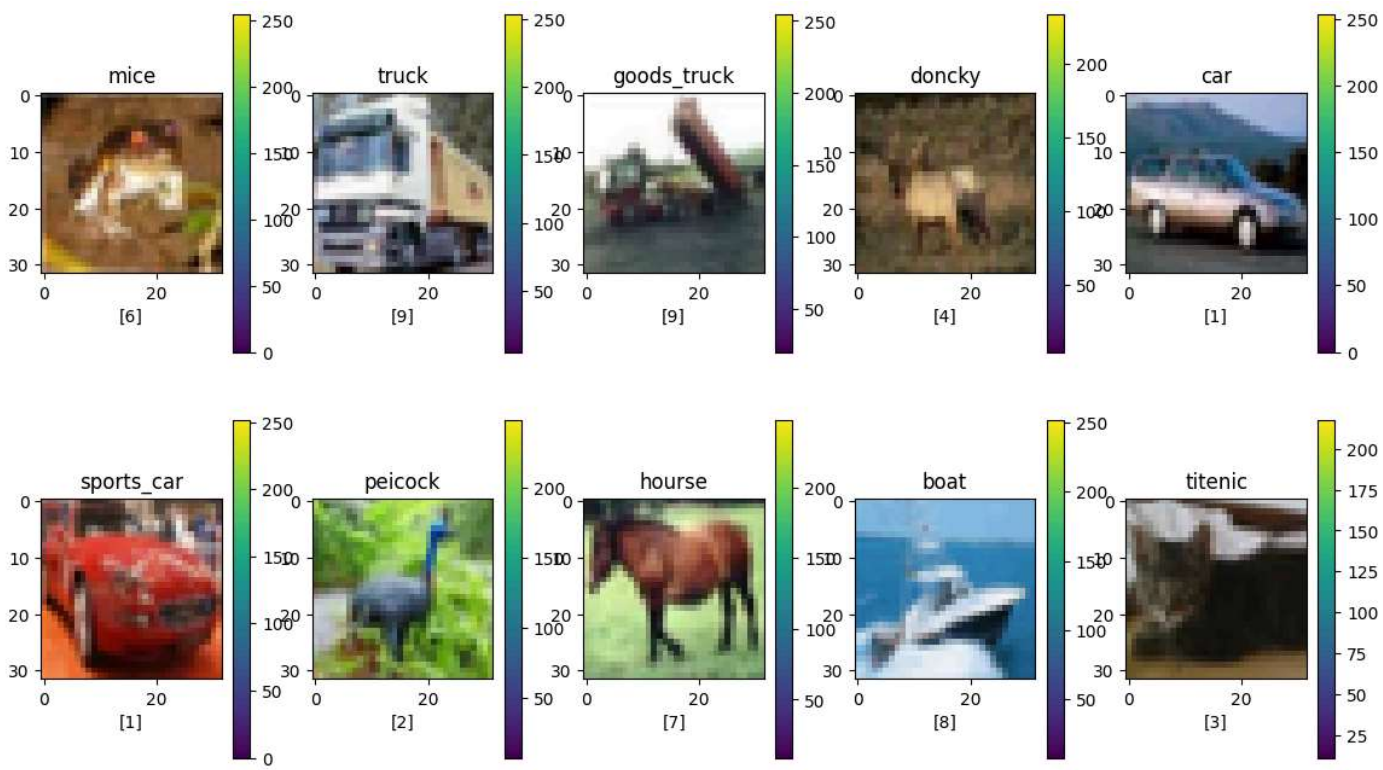
Step3:Create Class

```
class_names=["mice","truck","goods_truck","doncky","car","sports_car","peicock","hourse","boat","titenic"]
```

```
y_train
```

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

```
plt.figure(figsize=(14,8))
for i in range(10):
  plt.subplot(2,5,i+1)
  plt.imshow(x_train[i])
  plt.xlabel(y_train[i])
  plt.title(class_names[i])
  plt.colorbar()
plt.show()
```




```
x_train[0][1]
```

```
array([[ 16,  20,  20],
       [  0,   0,   0],
       [ 18,   8,   0],
       [ 51,  27,   8],
       [ 88,  51,  21],
       [120,  82,  43],
       [128,  89,  45],
       [127,  86,  44],
       [126,  87,  50],
       [116,  79,  44],
       [106,  70,  37],
       [101,  67,  35],
       [105,  70,  36],
       [113,  74,  35],
       [109,  70,  33],
       [112,  72,  37],
       [119,  79,  44],
       [109,  71,  33],
       [105,  69,  27],
       [125,  89,  46],
       [127,  92,  46],
       [122,  85,  39],
       [131,  89,  47],
       [124,  82,  41],
       [121,  79,  37],
       [131,  89,  48],
       [132,  91,  53],
       [133,  94,  58],
       [133,  96,  60],
       [123,  88,  55],
       [119,  83,  50],
       [122,  87,  57]], dtype=uint8)
```

```
class_names[y_train[0][0]]
```

```
'peicock'
```

Step4:Normalization

```
x_train=x_train/255.0
x_test=x_test/255.0
```

Create Neural Network

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Flatten

model=Sequential()

#input layer
model.add(Flatten(input_shape=(32,32,3)))

#hidden layer
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation='relu'))

#output layer
model.add(Dense(10,activation='softmax'))

model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim`
  super().__init__(**kwargs)
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 3072)	0
dense_8 (Dense)	(None, 128)	393,344
dense_9 (Dense)	(None, 64)	8,256
dense_10 (Dense)	(None, 32)	2,080
dense_11 (Dense)	(None, 10)	330

```
Total params: 404,010 (1.54 MB)
Trainable params: 404,010 (1.54 MB)
```

Step5:Model FIT

```
from re import VERBOSE

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

history=model.fit(x_train,
                  y_train,
                  epochs=10,
                  batch_size=64,
                  verbose=True)
```

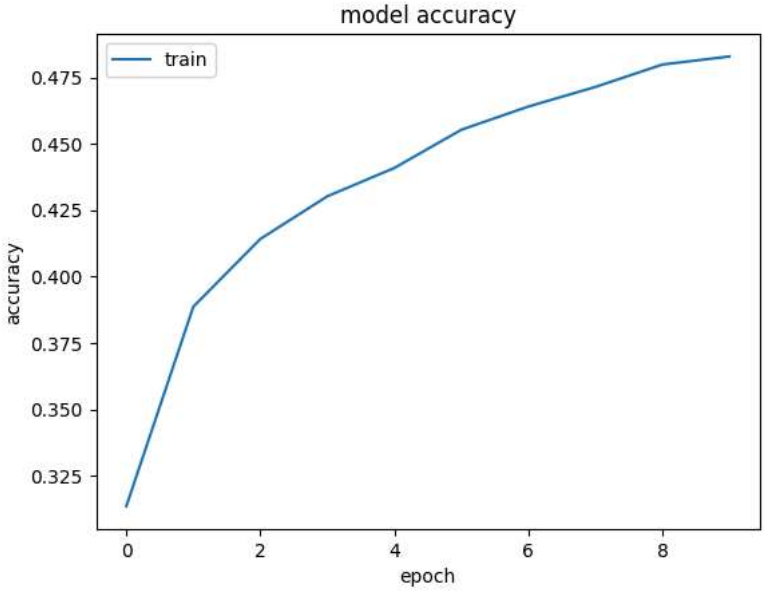
```
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 12s 13ms/step - accuracy: 0.2562 - loss: 2.0358
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.3796 - loss: 1.7192
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 8s 10ms/step - accuracy: 0.4138 - loss: 1.6415
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 9s 8ms/step - accuracy: 0.4300 - loss: 1.5988
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.4385 - loss: 1.5690
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.4553 - loss: 1.5245
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 8s 10ms/step - accuracy: 0.4624 - loss: 1.5061
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.4727 - loss: 1.4857
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 8s 10ms/step - accuracy: 0.4798 - loss: 1.4604
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.4823 - loss: 1.4534
```

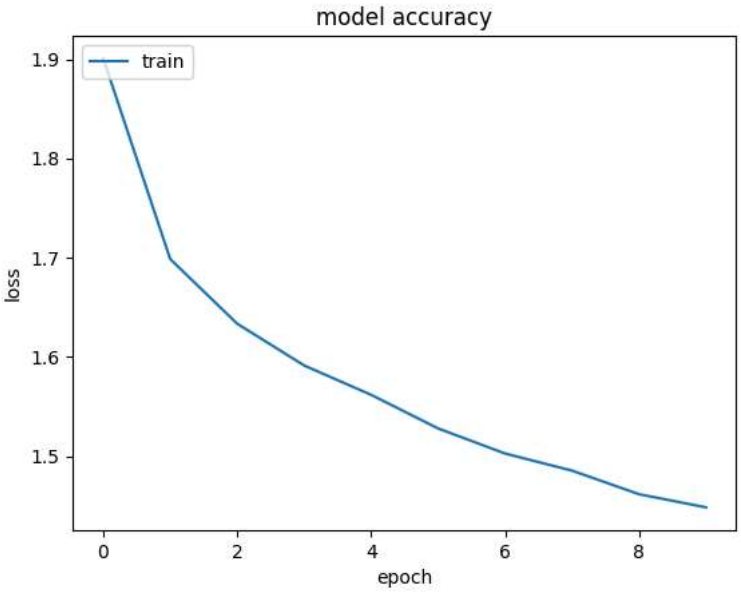
Step6:Model Evaluation

```
history.history

plt.plot(history.history['accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train'],loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.title('model accuracy')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train'],loc='upper left')
plt.show()
```






```
test_loss,test_accuracy=model.evaluate(x_test,y_test)
print('test accuracy',test_accuracy)
print('test loss',test_loss)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.4658 - loss: 1.4911
test accuracy 0.4645000100135803
test loss 1.4992070198059082
```

Step7:Model prediction

```
y_predict=model.predict(x_test)
y_predict
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
array([[0.07601365, 0.06522515, 0.11187036, ..., 0.01445887, 0.12085542,
        0.00609418],
       [0.04748309, 0.45226985, 0.00393258, ..., 0.00222319, 0.12607948,
        0.36089164],
       [0.24304488, 0.19261771, 0.016113  , ..., 0.01320632, 0.31085587,
        0.20582627],
       ...,
       [0.00377308, 0.00289733, 0.08369878, ..., 0.15148   , 0.01484672,
        0.00565735],
       [0.0095716 , 0.00293688, 0.14169939, ..., 0.12119258, 0.00562554,
        0.00388489],
       [0.06283853, 0.00538487, 0.12077242, ..., 0.4978586 , 0.00492396,
        0.02161144]], dtype=float32)
```

```
np.max(y_predict[0])
```

```
0.31284627
```

```
amx=np.max(y_predict[0])
amax=np.argmax(y_predict[0])
print(amx,amax,sep=",")
```

```
0.31284627,3
```

Step8:Prediction details

```
max_prob=np.max(y_predict,axis=1)

index=np.argmax(y_predict,axis=1)

predict_class=[class_names[i] for i in [i for i in index]]

Ground_truth_class=[class_names[i[0]] for i in [i for i in y_test]]
```

```
import pandas as pd
df=pd.DataFrame()
df["max prob"]=max prob
```