Surakshith Shetty -53026240013

4b. Predict the class in CIFAR 10 Dataset using CNN

```
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 ━━━━━━━━━━━━━━━━━━━━ 19s 0us/step
(50000, 32, 32, 3)
```

```
X_test.shape
```

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

```
y_train.shape
```

```
(50000, 1)
```

```
y_test.shape
```

```
(10000, 1)
```

```
X_train
```

```
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]]],
```

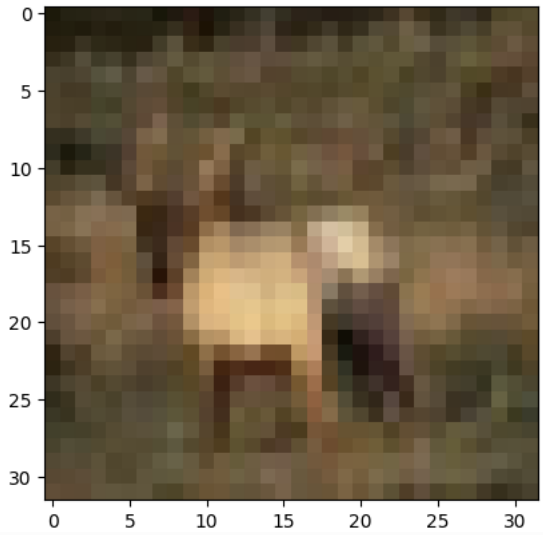
```
            [[[154, 177, 187],
              [126, 137, 136],
              [105, 104,  95],
              ...,
              [ 91,  95,  71],
              [ 87,  90,  71],
              [ 79,  81,  70]],
```

```
X_train[3]
```

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

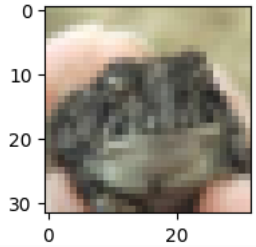
```
plt.imshow(X_train[3])
```

```
<matplotlib.image.AxesImage at 0x7ba0b0f63b50>
```




```
plt.figure(figsize=(15,2))
plt.imshow(X_train[25])
```

```
<matplotlib.image.AxesImage at 0x7ba0b0dab670>
```




```
y_train[:5]
```

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

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

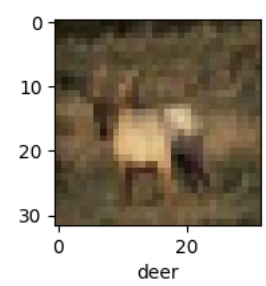
```
array([6, 9, 9, 4, 1], dtype=uint8)
```

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

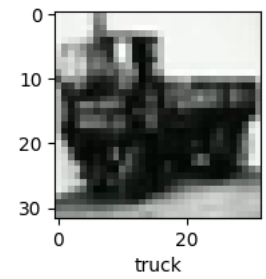
```
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

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

```
plot_sample(X_train, y_train,3)
```




```
plot_sample(X_train, y_train,205)
```




```
X_train[0]
```

ndarray (32, 32, 3) show data

```
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)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_c
  super().__init__(**kwargs)
Epoch 1/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.3061 - loss: 1.9332
Epoch 2/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - accuracy: 0.4184 - loss: 1.6453
Epoch 3/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.4491 - loss: 1.5586
Epoch 4/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.4723 - loss: 1.4874
Epoch 5/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.5003 - loss: 1.4307
<keras.src.callbacks.history.History at 0x7ba0b0ddfa30>
```

```
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 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.37      0.66      0.48      1000
           1       0.47      0.68      0.55      1000
           2       0.43      0.21      0.28      1000
           3       0.49      0.15      0.22      1000
```

```
           4       0.47      0.35      0.40      1000
           5       0.47      0.23      0.31      1000
           6       0.54      0.53      0.53      1000
           7       0.45      0.62      0.52      1000
           8       0.41      0.77      0.53      1000
           9       0.61      0.33      0.43      1000

    accuracy                           0.45     10000
   macro avg       0.47      0.45      0.43     10000
weighted avg       0.47      0.45      0.43     10000
```

```
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')
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
cnn.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])
cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.3780 - loss: 1.7068
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.5835 - loss: 1.1814
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.6383 - loss: 1.0295
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6733 - loss: 0.9360
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7004 - loss: 0.8638
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7177 - loss: 0.8102
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7358 - loss: 0.7592
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7472 - loss: 0.7242
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7608 - loss: 0.6818
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7784 - loss: 0.6390
<keras.src.callbacks.history.History at 0x7b9fd0d6cdc0>
```

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.6888 - loss: 0.9484
[0.9559218883514404, 0.6850000023841858]
```

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
array([[7.1856176e-04, 4.2133957e-05, 2.4897747e-03, 8.8845867e-01,
        9.7505159e-05, 8.2315199e-02, 4.0683406e-03, 1.2103402e-04,
        2.0653149e-02, 1.0355675e-03],
       [5.5850628e-03, 3.7936130e-01, 5.9351134e-07, 6.0526963e-06,
        5.7096532e-09, 6.8975986e-08, 1.5214437e-06, 2.4888963e-10,
        6.1479449e-01, 2.5089938e-04],
       [2.2185087e-02, 7.0236009e-01, 2.5795456e-03, 2.4339741e-03,
        2.0164954e-04, 4.3363700e-04, 1.2120159e-04, 1.0379347e-04,
        2.4175942e-01, 2.7821587e-02],
       [6.8832630e-01, 3.8397283e-04, 2.2262540e-01, 3.3243344e-04,
        8.5616339e-04, 9.1211677e-06, 2.6965374e-03, 7.2533177e-05,
        8.4651813e-02, 4.5721939e-05],
       [4.9037521e-05, 1.0783281e-04, 1.3226261e-02, 6.0132816e-02,
        7.1563566e-01, 5.7024523e-03, 2.0343499e-01, 9.1263710e-06,
        1.6686658e-03, 3.3204567e-05]], dtype=float32)
```