```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.datasets import cifar10

(xtrain,ytrain),(xtest,ytest)=cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-
python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
```

```
xtrain
```

```
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]],

        [[140, 160, 169],
         [145, 153, 154],
         [125, 125, 118],
         ...,
         [ 96,  99,  78],
         [ 77,  80,  62],
         [ 71,  73,  61]],

        [[140, 155, 164],
         [139, 146, 149],
         [115, 115, 112],
         ...,
         [ 79,  82,  64],
         [ 68,  70,  55],
         [ 67,  69,  55]],

        ...,

        [[175, 167, 166],
         [156, 154, 160],
         [154, 160, 170],
         ...,
         [ 42,  34,  36],
         [ 61,  53,  57],
         [ 93,  83,  91]],

        [[165, 154, 128],
         [156, 152, 130],
```

```
         [159, 161, 142],
         ...,
         [103,  93,  96],
         [123, 114, 120],
         [131, 121, 131]],

        [[163, 148, 120],
         [158, 148, 122],
         [163, 156, 133],
         ...,
         [143, 133, 139],
         [143, 134, 142],
         [143, 133, 144]]],


       [[[255, 255, 255],
         [253, 253, 253],
         [253, 253, 253],
         ...,
         [253, 253, 253],
         [253, 253, 253],
         [253, 253, 253]],

        [[255, 255, 255],
         [255, 255, 255],
         [255, 255, 255],
         ...,
         [255, 255, 255],
         [255, 255, 255],
         [255, 255, 255]],

        [[255, 255, 255],
         [254, 254, 254],
         [254, 254, 254],
         ...,
         [254, 254, 254],
         [254, 254, 254],
         [254, 254, 254]],

        ...,

        [[113, 120, 112],
         [111, 118, 111],
         [105, 112, 106],
         ...,
         [ 72,  81,  80],
         [ 72,  80,  79],
         [ 72,  80,  79]],

        [[111, 118, 110],
```

```
         [104, 111, 104],
         [ 99, 106,  98],
         ...,
         [ 68,  75,  73],
         [ 70,  76,  75],
         [ 78,  84,  82]],

        [[106, 113, 105],
         [ 99, 106,  98],
         [ 95, 102,  94],
         ...,
         [ 78,  85,  83],
         [ 79,  85,  83],
         [ 80,  86,  84]]],


       ...,


       [[[ 35, 178, 235],
         [ 40, 176, 239],
         [ 42, 176, 241],
         ...,
         [ 99, 177, 219],
         [ 79, 147, 197],
         [ 89, 148, 189]],

        [[ 57, 182, 234],
         [ 44, 184, 250],
         [ 50, 183, 240],
         ...,
         [156, 182, 200],
         [141, 177, 206],
         [116, 149, 175]],

        [[ 98, 197, 237],
         [ 64, 189, 252],
         [ 69, 192, 245],
         ...,
         [188, 195, 206],
         [119, 135, 147],
         [ 61,  79,  90]],

        ...,

        [[ 73,  79,  77],
         [ 53,  63,  68],
         [ 54,  68,  80],
         ...,
         [ 17,  40,  64],
```

```
         [ 21,  36,  51],
         [ 33,  48,  49]],

        [[ 61,  68,  75],
         [ 55,  70,  86],
         [ 57,  79, 103],
         ...,
         [ 24,  48,  72],
         [ 17,  35,  53],
         [  7,  23,  32]],

        [[ 44,  56,  73],
         [ 46,  66,  88],
         [ 49,  77, 105],
         ...,
         [ 27,  52,  77],
         [ 21,  43,  66],
         [ 12,  31,  50]]],


       [[[189, 211, 240],
         [186, 208, 236],
         [185, 207, 235],
         ...,
         [175, 195, 224],
         [172, 194, 222],
         [169, 194, 220]],

        [[194, 210, 239],
         [191, 207, 236],
         [190, 206, 235],
         ...,
         [173, 192, 220],
         [171, 191, 218],
         [167, 190, 216]],

        [[208, 219, 244],
         [205, 216, 240],
         [204, 215, 239],
         ...,
         [175, 191, 217],
         [172, 190, 216],
         [169, 191, 215]],

        ...,

        [[207, 199, 181],
         [203, 195, 175],
         [203, 196, 173],
         ...,
```

```
         [135, 132, 127],
         [162, 158, 150],
         [168, 163, 151]],

        [[198, 190, 170],
         [189, 181, 159],
         [180, 172, 147],
         ...,
         [178, 171, 160],
         [175, 169, 156],
         [175, 169, 154]],

        [[198, 189, 173],
         [189, 181, 162],
         [178, 170, 149],
         ...,
         [195, 184, 169],
         [196, 189, 171],
         [195, 190, 171]]],


       [[[229, 229, 239],
         [236, 237, 247],
         [234, 236, 247],
         ...,
         [217, 219, 233],
         [221, 223, 234],
         [222, 223, 233]],

        [[222, 221, 229],
         [239, 239, 249],
         [233, 234, 246],
         ...,
         [223, 223, 236],
         [227, 228, 238],
         [210, 211, 220]],

        [[213, 206, 211],
         [234, 232, 239],
         [231, 233, 244],
         ...,
         [220, 220, 232],
         [220, 219, 232],
         [202, 203, 215]],

        ...,

        [[150, 143, 135],
         [140, 135, 127],
         [132, 127, 120],
```

```
         ...,
         [224, 222, 218],
         [230, 228, 225],
         [241, 241, 238]],

        [[137, 132, 126],
         [130, 127, 120],
         [125, 121, 115],
         ...,
         [181, 180, 178],
         [202, 201, 198],
         [212, 211, 207]],

        [[122, 119, 114],
         [118, 116, 110],
         [120, 116, 111],
         ...,
         [179, 177, 173],
         [164, 164, 162],
         [163, 163, 161]]], dtype=uint8)
```

```
xtrain.shape,xtest.shape,ytrain.shape,ytest.shape
```

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

```
ytrain[:10]
```

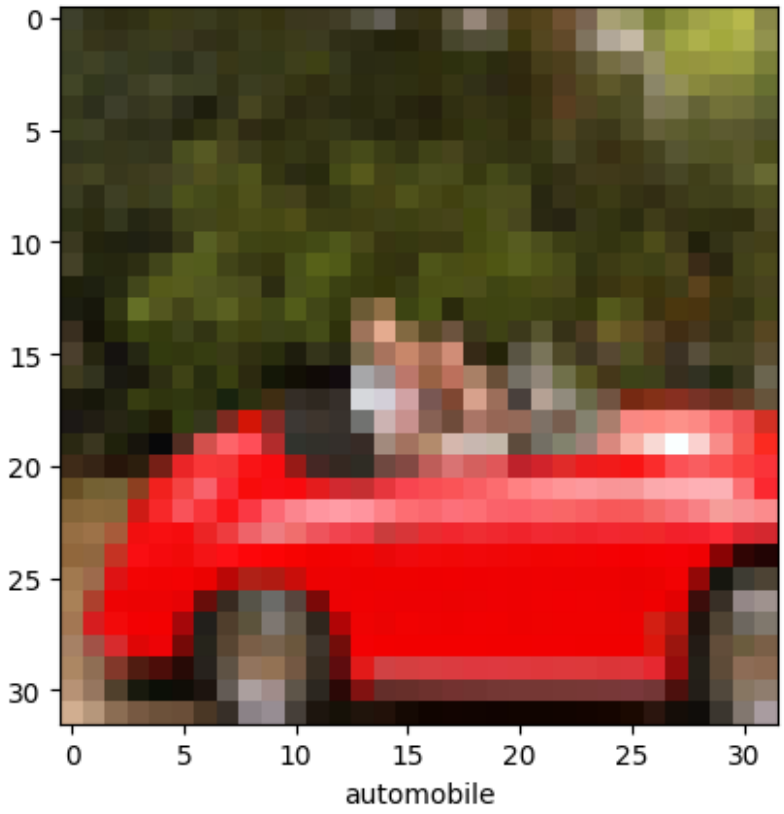
```
array([[6],
       [9],
       [9],
       [4],
       [1],
       [1],
       [2],
       [7],
       [8],
       [3]], dtype=uint8)
```

```
ytrain=ytrain.reshape(-1,)
ytest=ytest.reshape(-1,)
```

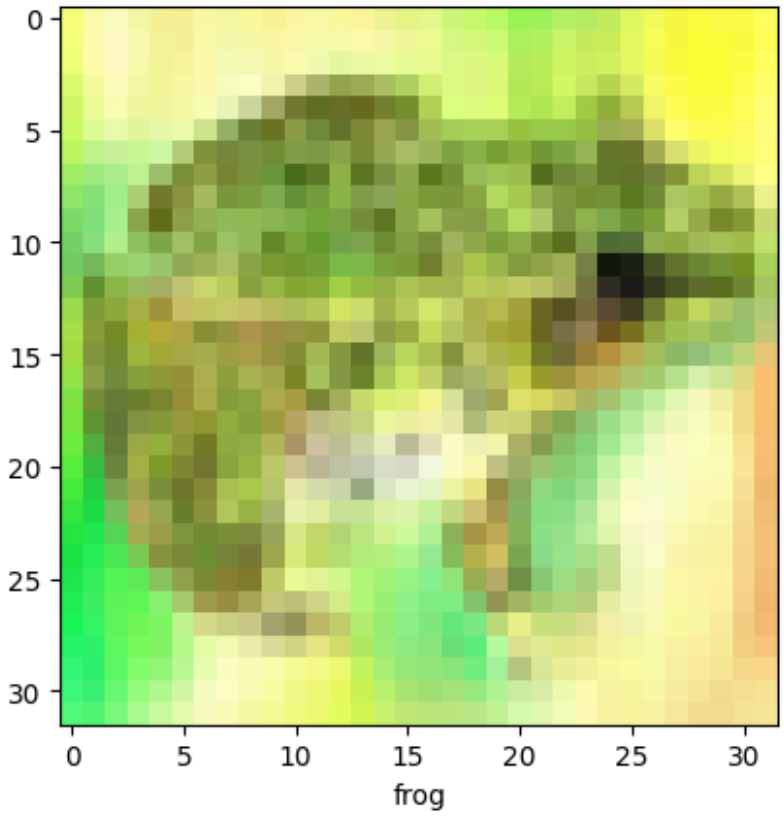
```
labels =
["airplane","automobile","bird","cat","deer","dog","frog","horse","shi
p","truck"]
labels
```

```
['airplane',
 'automobile',
 'bird',
 'cat',
 'deer',
```

```
 'dog',
 'frog',
 'horse',
 'ship',
 'truck']

def plot_pics(x, y, index):
    plt.imshow(x[index])
    plt.xlabel(labels[y[index]])

plot_pics(xtrain,ytrain,99)
```




```
plot_pics(xtrain,ytrain,95)
```




```
xtrain = xtrain.astype('float32') / 255.0
xtest = xtest.astype('float32') / 255.0

xtrain.dtype
```

```
dtype('float32')
```

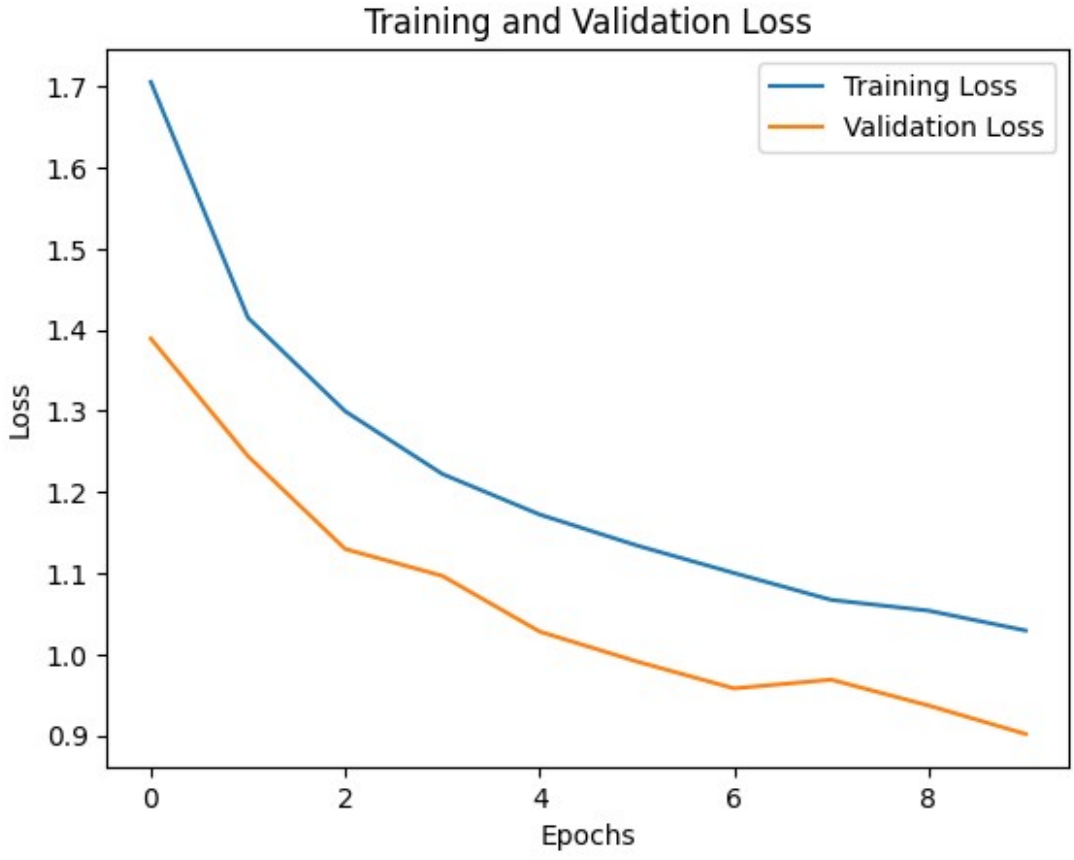
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD

 # Create an ImageDataGenerator object with augmentation parameters
datagen = ImageDataGenerator(
    rotation_range=15,       # Rotate images by up to 15 degrees
    width_shift_range=0.1,   # Shift images horizontally by up to 10%
of width
    height_shift_range=0.1,  # Shift images vertically by up to 10% of
height
    horizontal_flip=True,    # Randomly flip images horizontally
    zoom_range=0.2           # Randomly zoom into images by up to 20%
)
```

```
datagen.fit(xtrain)

ytrain = to_categorical(ytrain, 10)
ytest = to_categorical(ytest, 10)

ytrain.dtype

dtype('float64')

model=Sequential()

model.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(3
2,32,3)))
model.add(MaxPooling2D(2,2))

model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))

model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))

/usr/local/lib/python3.10/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(10,activation='softmax'))

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

fitting=model.fit(datagen.flow(xtrain, ytrain), epochs=10,
validation_data=(xtest, ytest))

Epoch 1/10

/usr/local/lib/python3.10/dist-packages/keras/src/trainers/
data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset`
class should call `super().__init__(**kwargs)` in its constructor.
`**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()

1563/1563 ━━━━━━━━━━━━━━━━━━━━ 113s 71ms/step - accuracy: 0.2885 -
loss: 1.9058 - val_accuracy: 0.5021 - val_loss: 1.3892
```

```
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 109s 70ms/step - accuracy: 0.4823 -
loss: 1.4456 - val_accuracy: 0.5593 - val_loss: 1.2442
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 115s 74ms/step - accuracy: 0.5261 -
loss: 1.3320 - val_accuracy: 0.6043 - val_loss: 1.1298
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 114s 73ms/step - accuracy: 0.5662 -
loss: 1.2295 - val_accuracy: 0.6071 - val_loss: 1.0967
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 113s 72ms/step - accuracy: 0.5855 -
loss: 1.1744 - val_accuracy: 0.6357 - val_loss: 1.0281
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 111s 71ms/step - accuracy: 0.5999 -
loss: 1.1331 - val_accuracy: 0.6460 - val_loss: 0.9912
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 113s 72ms/step - accuracy: 0.6095 -
loss: 1.1091 - val_accuracy: 0.6646 - val_loss: 0.9580
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 141s 71ms/step - accuracy: 0.6193 -
loss: 1.0739 - val_accuracy: 0.6559 - val_loss: 0.9688
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 112s 72ms/step - accuracy: 0.6250 -
loss: 1.0649 - val_accuracy: 0.6736 - val_loss: 0.9370
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 108s 69ms/step - accuracy: 0.6358 -
loss: 1.0340 - val_accuracy: 0.6831 - val_loss: 0.9018

model.evaluate(xtest,ytest)

313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step - accuracy: 0.6831 - loss:
0.8981

[0.9018288254737854, 0.6830999851226807]

ypred=model.predict(xtest)

313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step

ypred

array([[5.7629631e-03, 1.1925486e-02, 1.2597838e-02, ..., 1.2400440e-
02,
        1.2363757e-02, 4.5613926e-03],
       [1.7369345e-03, 3.7681642e-01, 8.9283576e-06, ..., 8.7462490e-
07,
        6.1920029e-01, 2.0550964e-03],
       [2.5143491e-02, 3.2987863e-01, 1.1793255e-03, ..., 1.8328280e-
04,
        5.6780428e-01, 6.8440385e-02],
       ...,
```

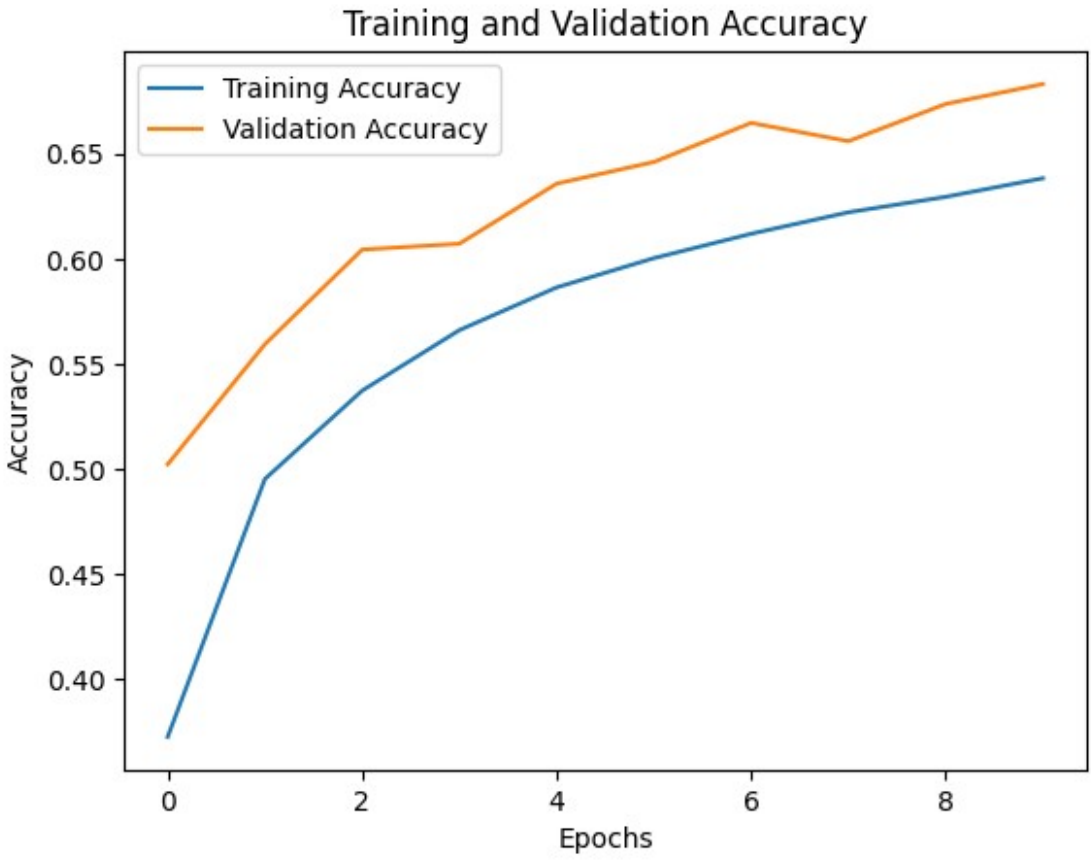
```
       [1.1167708e-03, 1.4920959e-04, 1.7325072e-02, ..., 2.4668530e-
02,
        2.2461796e-04, 9.2274195e-04],
       [1.0452790e-02, 9.5726103e-01, 7.5124092e-03, ..., 1.0882069e-
03,
        1.9493513e-04, 8.3943531e-03],
       [6.6534427e-05, 3.5955857e-06, 4.1101529e-04, ..., 9.3355352e-
01,
        2.1673354e-06, 9.3084418e-06]], dtype=float32)
```

```
plt.plot(fitting.history['loss'], label='Training Loss')
plt.plot(fitting.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')
plt.show()
```




```
plt.plot(fitting.history['accuracy'], label='Training Accuracy')
plt.plot(fitting.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
plt.legend()
plt.title('Training and Validation Accuracy')
plt.show()
```




```
ypred_labels = [np.argmax(element) for element in ypred]

ypred_labels
```

```
[5,
 8,
 8,
 0,
 4,
 6,
 1,
 6,
 3,
 1,
 0,
 9,
 5,
 7,
 9,
```

```
8,
5,
7,
8,
6,
7,
0,
2,
9,
4,
2,
4,
0,
9,
6,
6,
5,
4,
3,
9,
1,
7,
1,
9,
5,
0,
6,
7,
6,
0,
9,
3,
5,
7,
6,
9,
8,
7,
6,
8,
8,
5,
5,
3,
6,
7,
5,
6,
5,
```

```
6,
2,
1,
2,
3,
7,
0,
3,
8,
8,
9,
2,
0,
3,
5,
8,
8,
1,
1,
7,
2,
7,
2,
8,
9,
9,
0,
5,
8,
6,
4,
6,
6,
0,
0,
7,
7,
5,
6,
3,
1,
1,
2,
6,
3,
5,
7,
0,
2,
```

```
2,
1,
7,
0,
4,
3,
5,
8,
7,
1,
2,
8,
9,
0,
3,
6,
2,
4,
1,
8,
9,
1,
2,
9,
7,
6,
8,
4,
5,
6,
5,
8,
4,
6,
5,
5,
5,
8,
1,
6,
5,
0,
5,
5,
9,
3,
4,
0,
1,
```

```
1,
6,
8,
0,
8,
6,
5,
0,
9,
9,
1,
8,
1,
9,
3,
7,
3,
0,
0,
5,
2,
6,
5,
8,
6,
3,
3,
0,
6,
8,
2,
1,
7,
4,
8,
8,
3,
8,
5,
1,
8,
7,
1,
3,
8,
5,
7,
9,
5,
```

```
1,
5,
9,
1,
0,
7,
9,
0,
2,
7,
5,
9,
5,
3,
9,
7,
0,
5,
6,
5,
1,
3,
8,
8,
0,
4,
0,
5,
3,
1,
1,
8,
9,
0,
3,
1,
8,
2,
8,
5,
5,
9,
9,
2,
8,
6,
0,
8,
1,
```

```
8,
9,
5,
3,
8,
8,
5,
4,
7,
2,
4,
3,
6,
5,
8,
7,
0,
1,
7,
5,
9,
0,
3,
1,
9,
9,
1,
8,
7,
9,
1,
2,
6,
1,
3,
2,
6,
0,
2,
6,
6,
6,
5,
5,
2,
1,
8,
2,
1,
```

```
2,
1,
6,
0,
2,
3,
7,
7,
7,
5,
5,
3,
3,
6,
2,
5,
7,
7,
7,
4,
6,
1,
9,
3,
6,
6,
9,
3,
8,
2,
7,
4,
6,
0,
5,
8,
5,
7,
6,
8,
1,
9,
1,
8,
2,
6,
3,
5,
2,
```

```
1,
0,
9,
5,
8,
1,
9,
7,
1,
1,
2,
1,
2,
7,
9,
4,
2,
7,
2,
7,
8,
6,
6,
9,
0,
1,
5,
0,
7,
2,
3,
5,
1,
2,
6,
5,
9,
6,
5,
1,
0,
1,
9,
8,
7,
8,
8,
6,
0,
```

```
1,
8,
2,
7,
0,
5,
6,
1,
9,
0,
7,
5,
7,
4,
3,
0,
1,
2,
9,
3,
4,
6,
6,
4,
5,
3,
7,
9,
7,
1,
5,
5,
9,
1,
2,
9,
9,
5,
4,
5,
0,
2,
2,
3,
9,
7,
3,
9,
5,
```

```
6,
5,
6,
2,
5,
6,
1,
5,
7,
4,
7,
5,
7,
8,
4,
7,
8,
8,
7,
7,
4,
0,
5,
7,
9,
6,
8,
5,
5,
0,
9,
9,
4,
0,
1,
0,
0,
1,
1,
8,
0,
5,
2,
0,
6,
6,
5,
4,
9,
4,
```

```
7,
9,
3,
4,
5,
6,
6,
1,
5,
5,
8,
1,
3,
8,
5,
7,
0,
7,
0,
3,
4,
0,
0,
6,
9,
5,
1,
5,
6,
6,
6,
6,
8,
9,
1,
7,
6,
7,
5,
9,
1,
6,
6,
5,
5,
5,
8,
5,
9,
```

```
4,
6,
3,
5,
2,
0,
7,
6,
3,
2,
5,
9,
3,
6,
4,
6,
7,
1,
5,
6,
6,
8,
8,
7,
5,
4,
0,
8,
1,
0,
8,
5,
4,
8,
9,
6,
1,
2,
6,
1,
4,
7,
5,
5,
3,
9,
5,
0,
2,
```

```
1,
6,
4,
5,
3,
9,
6,
9,
8,
8,
5,
8,
6,
6,
7,
1,
7,
7,
1,
5,
7,
9,
9,
4,
7,
8,
8,
2,
6,
8,
7,
6,
8,
8,
0,
5,
5,
6,
7,
7,
9,
1,
3,
4,
4,
7,
3,
9,
9,
```

```
6,
9,
2,
1,
1,
4,
1,
9,
7,
7,
2,
3,
0,
9,
0,
1,
2,
6,
7,
5,
3,
2,
1,
4,
9,
5,
5,
9,
6,
2,
9,
7,
2,
9,
6,
6,
6,
3,
2,
2,
7,
8,
3,
8,
2,
7,
5,
7,
2,
```

```
3,
8,
7,
4,
2,
1,
0,
8,
6,
0,
8,
7,
4,
5,
3,
8,
4,
9,
7,
8,
8,
1,
8,
1,
1,
5,
6,
5,
3,
2,
7,
9,
1,
2,
1,
0,
1,
7,
5,
7,
0,
7,
5,
7,
6,
6,
2,
3,
9,
```

```
0,
9,
1,
2,
2,
6,
8,
2,
1,
3,
5,
1,
0,
1,
2,
5,
9,
5,
4,
7,
9,
6,
3,
0,
4,
5,
6,
0,
6,
9,
1,
7,
6,
7,
8,
3,
9,
6,
8,
0,
0,
7,
7,
4,
7,
4,
6,
2,
2,
```

```
7,
1,
2,
7,
4,
4,
8,
6,
7,
5,
2,
5,
7,
2,
9,
8,
9,
3,
8,
1,
6,
5,
0,
8,
7,
6,
7,
6,
7,
3,
1,
6,
2,
5,
2,
3,
1,
2,
2,
2,
7,
0,
7,
2,
1,
0,
2,
0,
6,
```

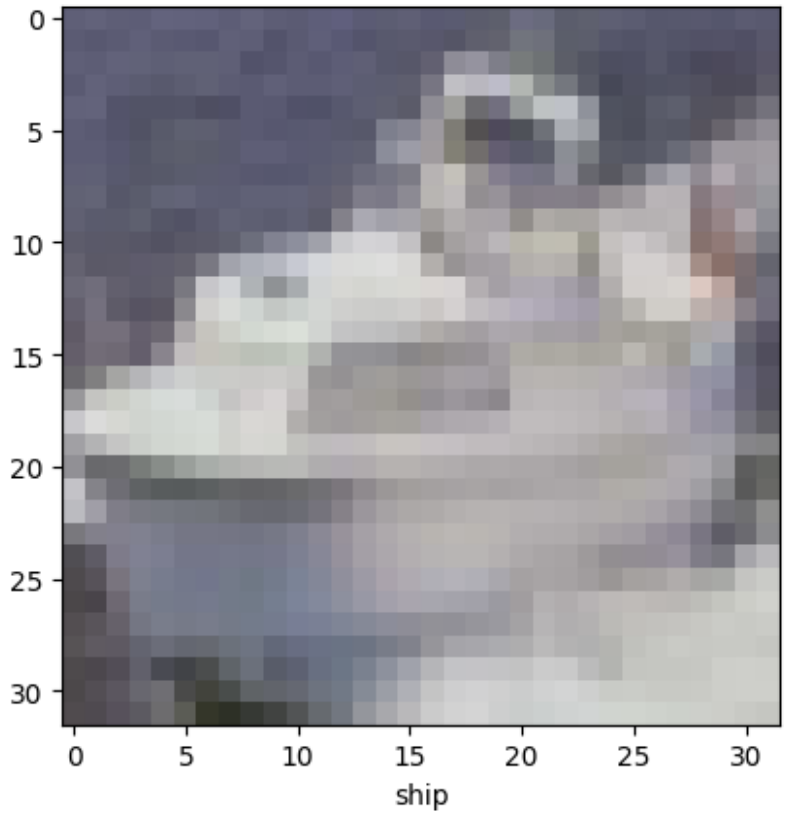
```
4,
5,
1,
8,
1,
0,
7,
7,
0,
7,
5,
4,
6,
6,
5,
0,
1,
1,
7,
0,
1,
3,
1,
4,
2,
3,
8,
4,
5,
2,
7,
8,
3,
0,
0,
9,
0,
0,
1,
0,
6,
3,
6,
7,
5,
1,
1,
6,
7,
```

```
7,
5,
6,
6,
6,
5,
0,
5,
1,
6,
8,
8,
5,
3,
2,
4,
0,
1,
2,
8,
8,
2,
6,
9,
9,
9,
1,
3,
8,
6,
0,
8,
4,
2,
3,
5,
7,
3,
2,
5,
1,
8,
9,
1,
7,
6,
0,
3,
0,
```

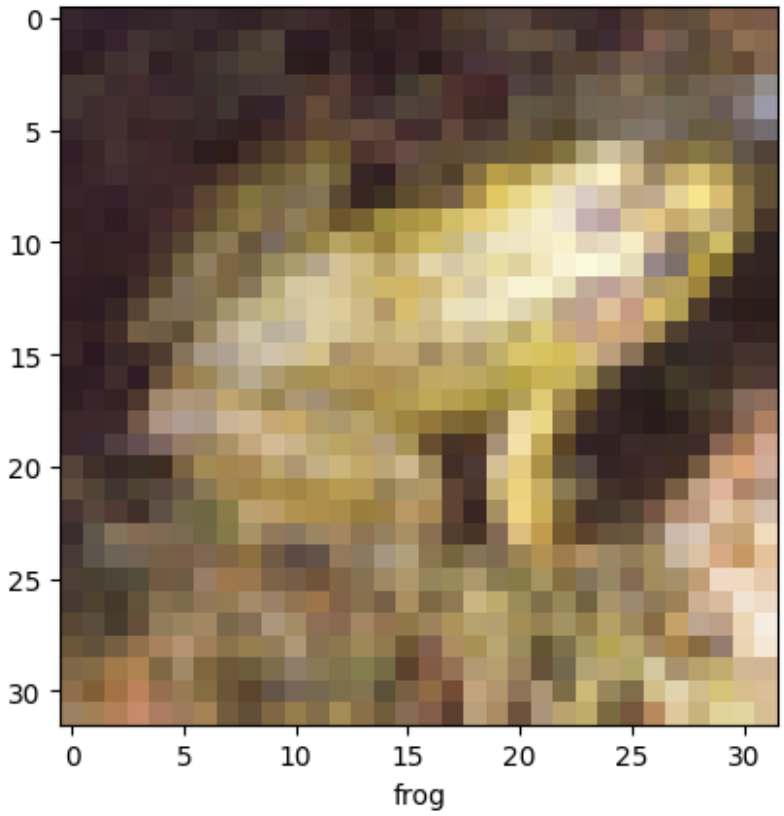
```
1,
3,
8,
6,
9,
6,
1,
4,
7,
2,
6,
7,
8,
9,
1,
1,
6,
3,
5,
6,
9,
1,
9,
1,
4,
6,
1,
2,
0,
6,
8,
1,
9,
6,
1,
5,
4,
7,
0,
5,
1,
2,
0,
1,
6,
8,
5,
6,
5,
8,
```

```
 1,
 3,
 8,
 ...]
```

```
plot_pics(xtest,ypred_labels,199)
```




```
plot_pics(xtest,ypred_labels,19)
```




```
model1=Sequential()

model1.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(
32,32,3),kernel_regularizer=regularizers.l2(0.001)))
model1.add(MaxPooling2D(2,2))

model1.add(Conv2D(64,kernel_size=(3,3),activation='relu',kernel_regula
rizer=regularizers.l2(0.001)))
model1.add(MaxPooling2D(2,2))

model1.add(Conv2D(128,kernel_size=(3,3),activation='relu',kernel_regul
arizer=regularizers.l2(0.001)))
model1.add(MaxPooling2D(2,2))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
model1.add(Flatten())
model1.add(Dense(64,activation='relu',kernel_regularizer=regularizers.
```

```
l2(0.001)))
model1.add(Dense(10,activation='softmax'))

model1.compile(optimizer=SGD(learning_rate=0.01),loss='categorical_cro
ssentropy',metrics=['accuracy'])

fitting_l2=model1.fit(datagen.flow(xtrain, ytrain), epochs=10,
validation_data=(xtest, ytest))
```

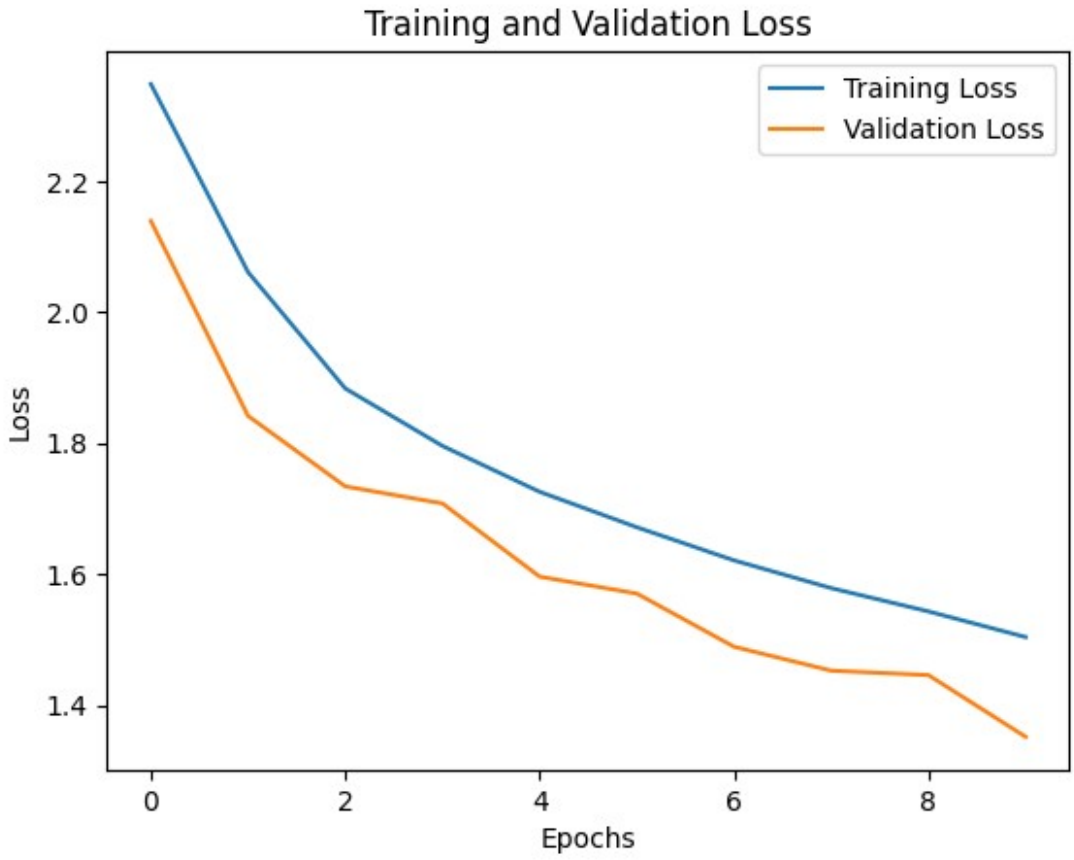
```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 116s 73ms/step - accuracy: 0.1827 -
loss: 2.4517 - val_accuracy: 0.3030 - val_loss: 2.1395
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 117s 75ms/step - accuracy: 0.3179 -
loss: 2.1186 - val_accuracy: 0.4250 - val_loss: 1.8417
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 115s 74ms/step - accuracy: 0.3926 -
loss: 1.9114 - val_accuracy: 0.4628 - val_loss: 1.7340
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 118s 75ms/step - accuracy: 0.4257 -
loss: 1.8201 - val_accuracy: 0.4668 - val_loss: 1.7080
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 139s 73ms/step - accuracy: 0.4518 -
loss: 1.7419 - val_accuracy: 0.4979 - val_loss: 1.5961
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 115s 74ms/step - accuracy: 0.4714 -
loss: 1.6871 - val_accuracy: 0.5123 - val_loss: 1.5703
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 119s 76ms/step - accuracy: 0.4925 -
loss: 1.6298 - val_accuracy: 0.5440 - val_loss: 1.4890
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 116s 74ms/step - accuracy: 0.5045 -
loss: 1.5877 - val_accuracy: 0.5545 - val_loss: 1.4525
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 142s 74ms/step - accuracy: 0.5184 -
loss: 1.5544 - val_accuracy: 0.5589 - val_loss: 1.4457
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 144s 75ms/step - accuracy: 0.5363 -
loss: 1.5144 - val_accuracy: 0.5876 - val_loss: 1.3514
```

```
model1.evaluate(xtest, ytest)
```

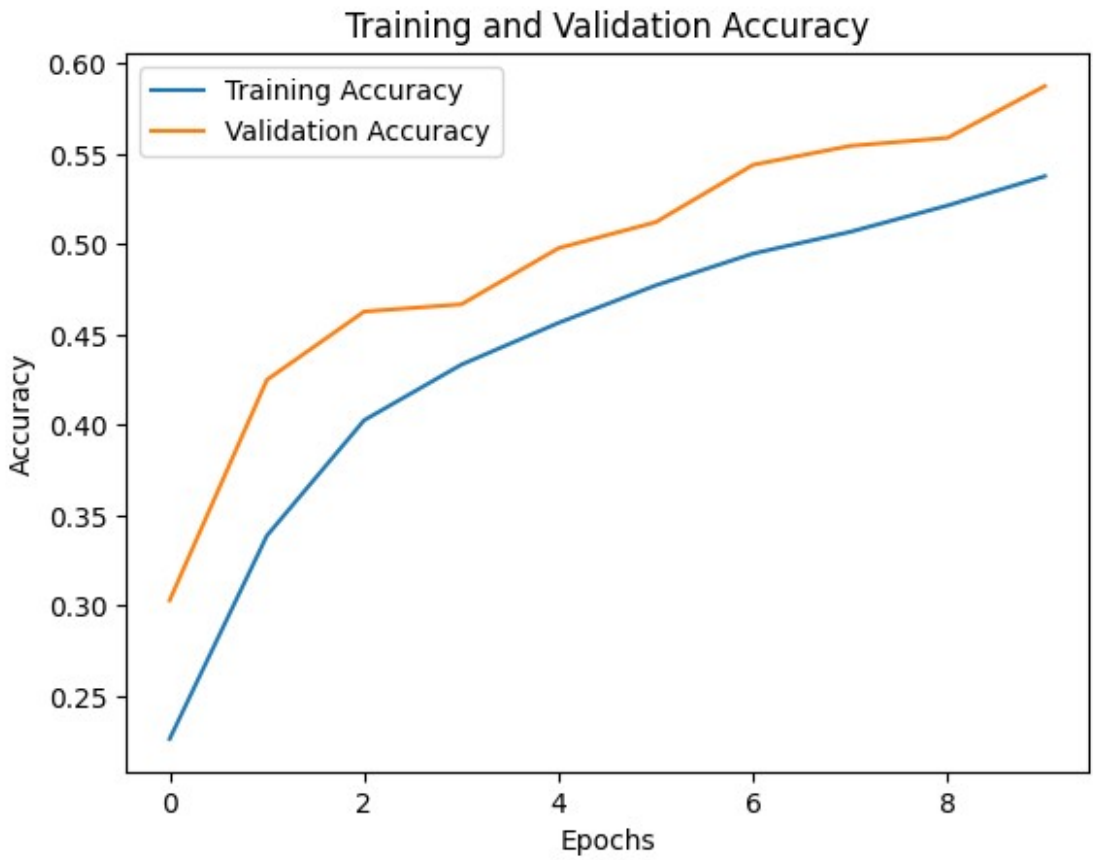
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 14ms/step - accuracy: 0.5892 - loss:
1.3449

[1.3514326810836792, 0.5875999927520752]
```

```
plt.plot(fitting_l2.history['loss'], label='Training Loss')
plt.plot(fitting_l2.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.legend()
plt.title('Training and Validation Loss')
plt.show()
```




```
plt.plot(fitting_l2.history['accuracy'], label='Training Accuracy')
plt.plot(fitting_l2.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')
plt.show()
```




```
ypred2=model1.predict(xtest)
ypred2
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step
```

```
array([[3.36247310e-02, 9.28371958e-03, 7.90022239e-02, ...,
        4.23690025e-03, 8.11516047e-02, 5.61624160e-03],
       [6.91482574e-02, 6.47427440e-01, 5.09673031e-04, ...,
        1.06966305e-04, 2.38031164e-01, 4.42746133e-02],
       [6.43365234e-02, 4.14478987e-01, 2.44213874e-03, ...,
        2.01703678e-03, 3.54714751e-01, 1.55825496e-01],
       ...,
       [1.13222760e-03, 5.51348901e-04, 4.96408418e-02, ...,
        1.22194611e-01, 1.11980911e-03, 4.18143300e-03],
       [1.33047804e-01, 2.26266608e-01, 8.04703161e-02, ...,
        1.94843024e-01, 1.28673967e-02, 1.68468524e-02],
       [5.59899665e-04, 1.09753374e-03, 2.69893534e-03, ...,
        9.36102271e-01, 1.80285846e-04, 3.22660501e-03]],
dtype=float32)
```

```
ypred_labels1 = [np.argmax(element) for element in ypred2]
```

```
ypred_labels1
```

```
[3,
 1,
 1,
 0,
 4,
 6,
 1,
 6,
 5,
 1,
 8,
 9,
 5,
 7,
 1,
 6,
 5,
 5,
 8,
 6,
 7,
 0,
 0,
 9,
 2,
 4,
 4,
 7,
 1,
 6,
 6,
 5,
 4,
 3,
 9,
 1,
 4,
 1,
 9,
 5,
 1,
 6,
 5,
 6,
 0,
 9,
 3,
 3,
 7,
 6,
```

```
9,
8,
6,
3,
8,
8,
7,
5,
3,
4,
7,
3,
1,
0,
6,
6,
1,
0,
5,
7,
4,
5,
8,
8,
9,
4,
9,
3,
5,
8,
8,
1,
1,
7,
2,
7,
2,
8,
8,
1,
0,
5,
8,
6,
4,
6,
6,
8,
0,
7,
```

```
4,
4,
6,
3,
1,
1,
2,
6,
5,
5,
4,
0,
2,
2,
9,
3,
0,
3,
6,
5,
8,
7,
1,
2,
8,
9,
8,
3,
3,
0,
4,
1,
8,
9,
1,
3,
9,
7,
2,
8,
4,
5,
6,
7,
1,
3,
6,
6,
5,
3,
```

```
1,
1,
6,
8,
0,
5,
3,
9,
3,
4,
2,
1,
1,
4,
8,
7,
8,
4,
9,
7,
9,
6,
1,
8,
9,
9,
6,
7,
3,
9,
8,
7,
6,
8,
4,
8,
6,
3,
3,
8,
5,
5,
0,
1,
7,
4,
8,
8,
3,
8,
```

```
3,
1,
8,
7,
1,
3,
8,
5,
7,
9,
3,
1,
3,
9,
8,
4,
0,
9,
0,
7,
7,
6,
9,
5,
5,
9,
3,
4,
4,
1,
5,
1,
5,
1,
8,
0,
4,
0,
8,
3,
1,
1,
8,
9,
0,
8,
4,
8,
2,
0,
```

```
5,
3,
9,
9,
4,
8,
6,
0,
8,
1,
8,
8,
3,
3,
0,
8,
8,
4,
7,
0,
4,
2,
6,
5,
8,
3,
1,
2,
7,
5,
8,
1,
6,
1,
9,
9,
1,
8,
7,
9,
1,
2,
6,
1,
5,
2,
6,
0,
0,
6,
```

```
6,
4,
7,
8,
2,
8,
8,
2,
1,
2,
8,
6,
0,
0,
3,
0,
7,
7,
5,
5,
5,
5,
6,
4,
5,
9,
7,
3,
8,
6,
8,
9,
4,
6,
6,
6,
6,
8,
0,
1,
5,
6,
0,
1,
8,
5,
7,
5,
8,
9,
```

```
1,
1,
8,
0,
6,
5,
4,
3,
1,
0,
9,
5,
8,
8,
9,
7,
1,
1,
0,
1,
2,
7,
9,
4,
2,
7,
8,
7,
8,
6,
6,
9,
0,
8,
5,
0,
7,
4,
3,
5,
1,
0,
6,
7,
9,
6,
2,
3,
8,
1,
```

```
9,
1,
7,
8,
8,
6,
0,
1,
8,
2,
5,
0,
5,
6,
1,
9,
4,
7,
5,
7,
4,
5,
8,
0,
2,
9,
5,
5,
6,
6,
2,
5,
5,
7,
9,
7,
1,
4,
5,
9,
1,
4,
9,
8,
5,
4,
5,
0,
2,
2,
```

```
6,
9,
7,
3,
9,
4,
2,
5,
6,
8,
2,
6,
1,
5,
3,
4,
7,
5,
7,
8,
5,
3,
8,
8,
3,
7,
4,
8,
5,
4,
9,
6,
8,
7,
6,
9,
9,
9,
5,
0,
1,
9,
8,
1,
8,
8,
0,
2,
2,
0,
```

```
4,
6,
5,
6,
9,
2,
3,
9,
8,
4,
5,
6,
6,
1,
5,
3,
8,
9,
9,
8,
5,
7,
0,
7,
0,
3,
4,
0,
8,
6,
8,
9,
6,
0,
6,
6,
6,
6,
8,
8,
1,
7,
6,
7,
5,
9,
1,
6,
6,
8,
```

```
2,
5,
8,
3,
9,
8,
6,
3,
3,
2,
0,
7,
6,
2,
6,
5,
9,
3,
9,
6,
6,
7,
1,
8,
3,
6,
8,
8,
7,
3,
4,
0,
8,
1,
0,
8,
3,
4,
8,
1,
6,
1,
2,
6,
1,
4,
7,
5,
5,
```

```
6,
8,
5,
0,
2,
1,
6,
5,
5,
3,
9,
6,
9,
8,
8,
5,
8,
1,
6,
7,
1,
7,
2,
1,
2,
7,
9,
9,
4,
7,
5,
8,
2,
6,
3,
7,
6,
9,
4,
8,
2,
3,
6,
1,
7,
9,
1,
3,
2,
```

```
2,
5,
3,
9,
9,
6,
9,
0,
1,
1,
4,
1,
9,
7,
7,
4,
3,
8,
1,
0,
9,
2,
6,
5,
5,
6,
2,
1,
4,
9,
3,
3,
9,
6,
8,
8,
9,
4,
9,
5,
6,
6,
3,
2,
0,
7,
8,
3,
8,
```

```
4,
7,
5,
7,
8,
5,
8,
7,
6,
8,
5,
8,
8,
3,
0,
8,
7,
5,
3,
4,
8,
4,
8,
9,
8,
8,
1,
8,
1,
1,
2,
6,
3,
5,
0,
7,
1,
1,
4,
1,
0,
1,
7,
5,
7,
8,
7,
2,
7,
```

```
6,
6,
2,
3,
9,
0,
9,
1,
2,
4,
6,
8,
8,
1,
3,
4,
1,
0,
1,
2,
3,
9,
3,
4,
1,
9,
6,
5,
0,
4,
3,
3,
0,
6,
9,
1,
7,
6,
9,
8,
6,
9,
2,
8,
0,
0,
3,
9,
7,
```

```
7,
2,
6,
3,
4,
7,
1,
2,
7,
4,
4,
8,
4,
2,
8,
5,
3,
7,
4,
1,
8,
9,
3,
1,
2,
8,
2,
8,
8,
7,
3,
4,
6,
5,
0,
1,
3,
2,
5,
2,
3,
8,
7,
9,
2,
7,
0,
7,
4,
```

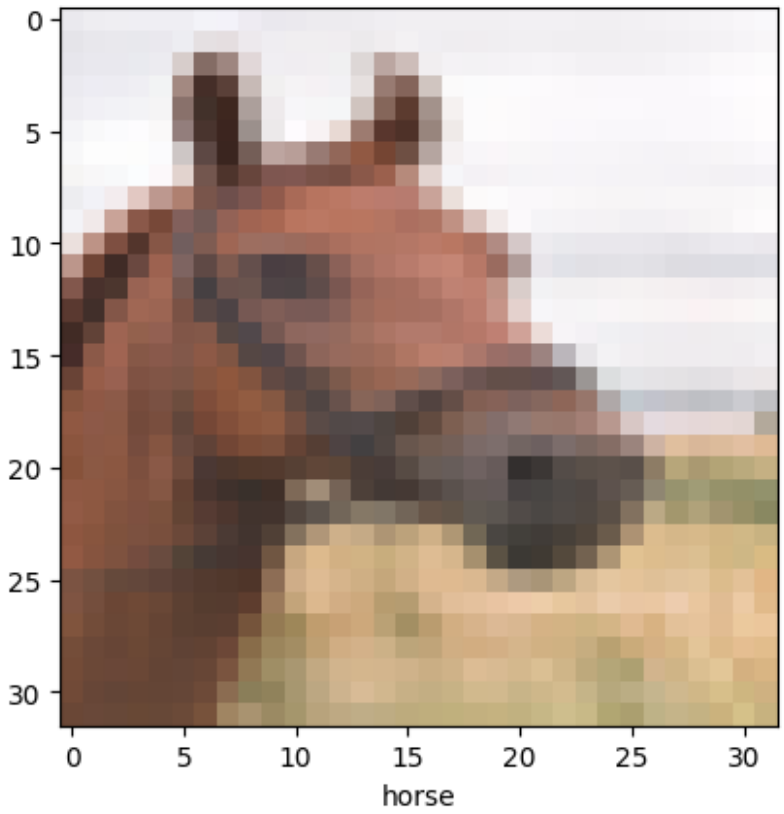
```
1,
8,
4,
9,
6,
4,
5,
9,
8,
1,
8,
9,
7,
0,
7,
8,
4,
4,
6,
0,
0,
1,
9,
7,
0,
1,
3,
1,
4,
8,
3,
8,
4,
3,
8,
7,
8,
4,
0,
0,
9,
0,
8,
1,
8,
6,
3,
6,
5,
```

```
3,
1,
1,
6,
7,
5,
5,
4,
3,
6,
5,
8,
0,
1,
6,
8,
8,
5,
3,
3,
4,
0,
9,
2,
8,
8,
4,
6,
9,
9,
6,
1,
5,
6,
2,
0,
8,
4,
2,
3,
3,
7,
2,
3,
7,
1,
8,
1,
7,
```

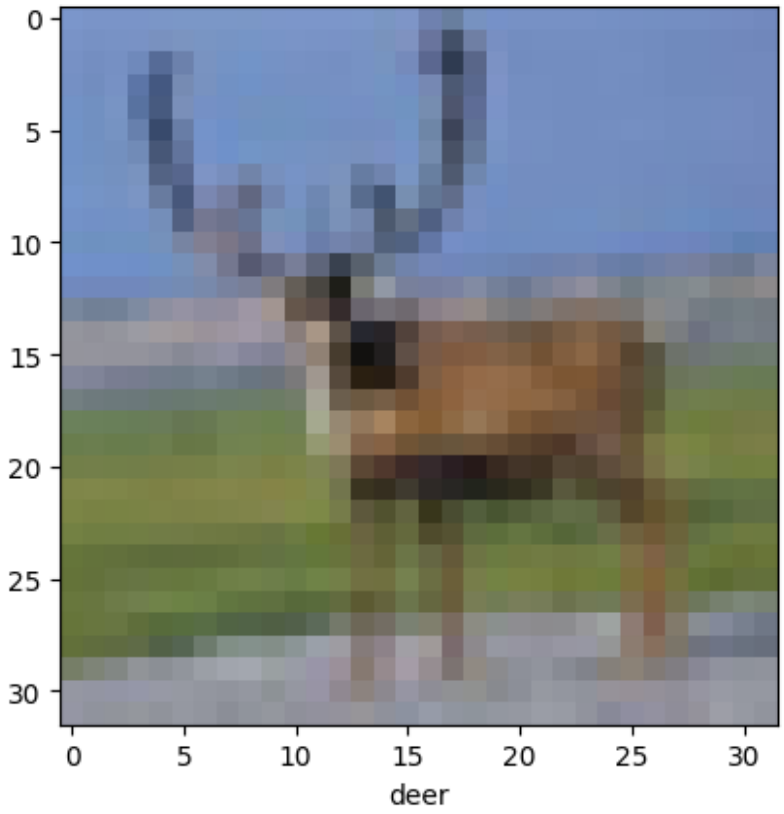
```
7,
4,
8,
5,
0,
1,
5,
8,
3,
9,
4,
1,
4,
7,
0,
5,
7,
8,
9,
1,
1,
6,
5,
7,
9,
9,
1,
9,
9,
4,
2,
1,
0,
0,
6,
8,
1,
1,
6,
1,
5,
4,
7,
8,
5,
1,
5,
0,
1,
```

```
 3,
 8,
 7,
 6,
 5,
 8,
 8,
 5,
 8,
 ...]
```

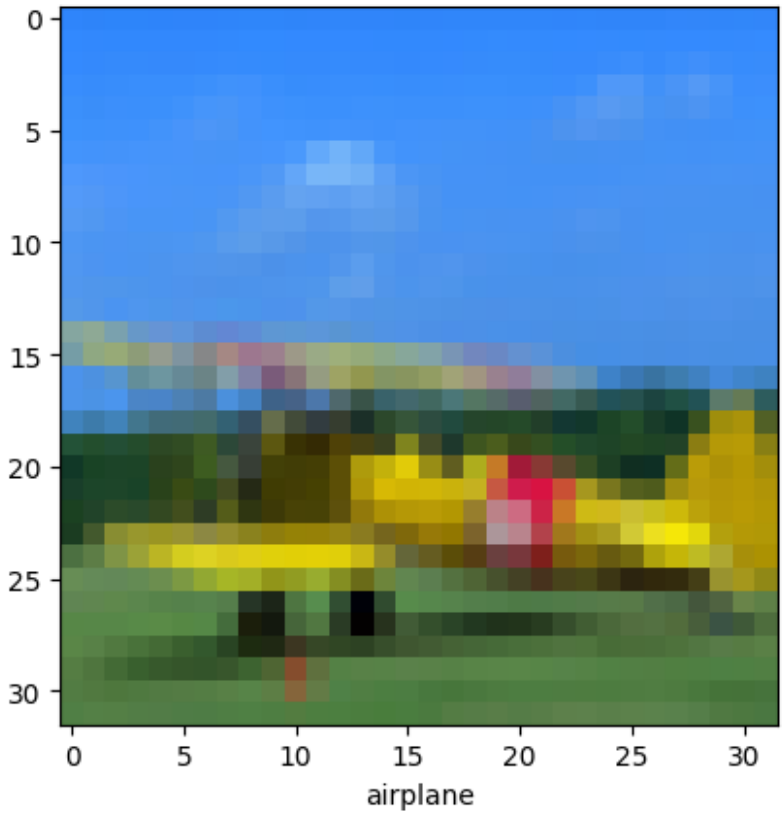
```
plot_pics(xtest,ypred_labels1,99)
```



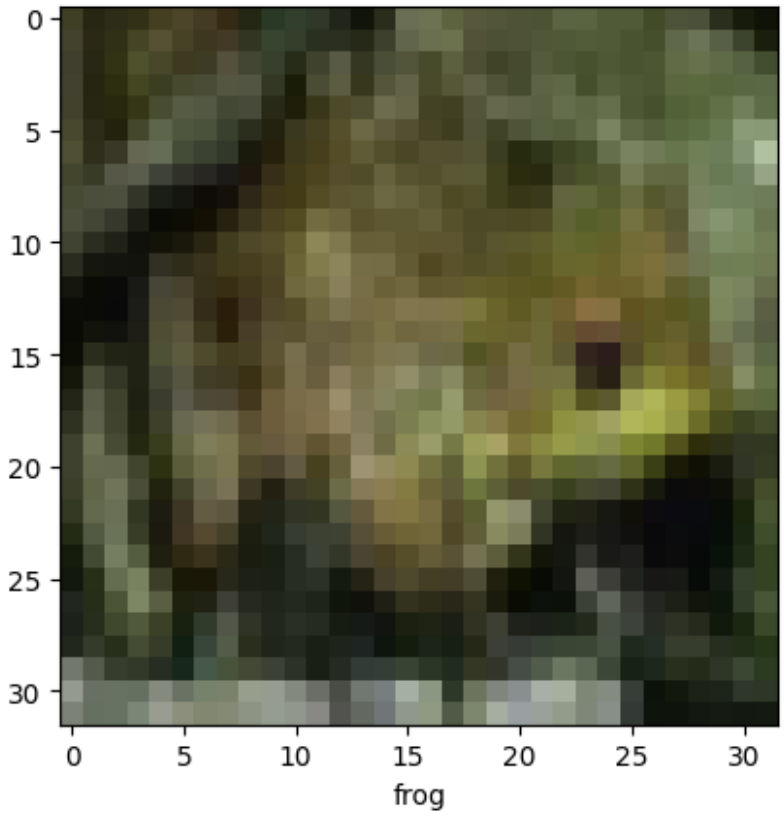

```
plot_pics(xtest,ypred_labels1,159)
```



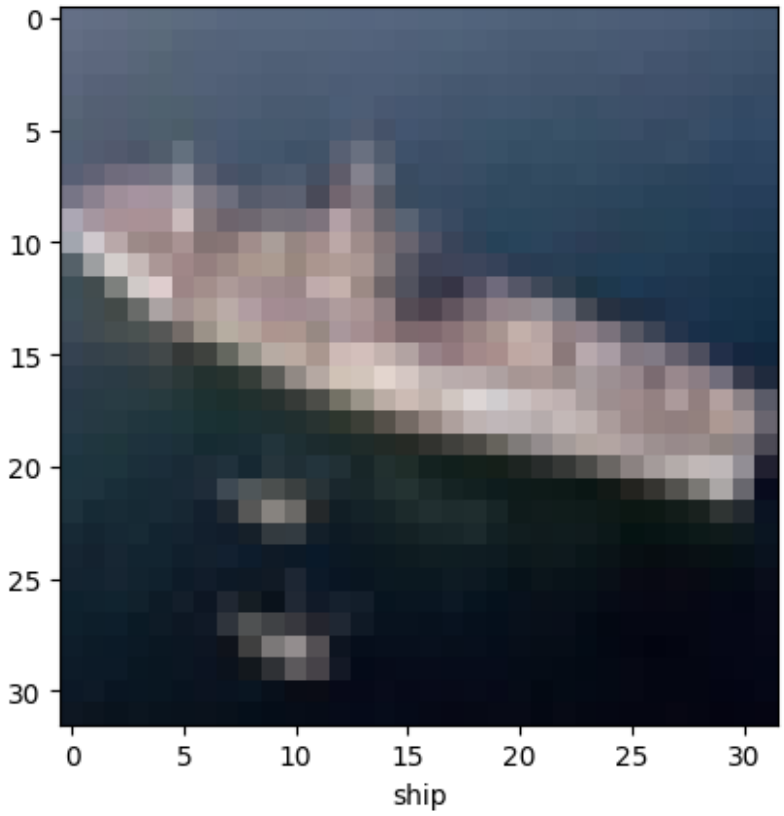

```
plot_pics(xtest,ypred_labels1,44)
```




```
plot_pics(xtest,ypred_labels1,107)
```




```
plot_pics(xtest,ypred_labels1,789)
```


0
5
10
15
20
25
30
0
5
10
15
20
25
30
ship