```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

```
from tensorflow.keras.datasets import cifar100
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
169001437/169001437 ━━━━━━━━━━━━━━━━━━━━ 11s 0us/step
```

```
x_train.shape
```

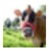
```
(50000, 32, 32, 3)
```

```
x_test.shape
```

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

```
x_train[0]
```

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



```
print(x_train[0])
```

```
[[[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [195 205 193]
  [212 224 204]
  [182 194 167]]

 [[255 255 255]
  [254 254 254]
  [254 254 254]
  ...
  [170 176 150]
  [161 168 130]
  [146 154 113]]

 [[255 255 255]
  [254 254 254]
  [255 255 255]
  ...
  [189 199 169]
  [166 178 130]
  [121 133  87]]

 ...

 [[148 185  79]
  [142 182  57]
  [140 179  60]
  ...
  [ 30  17   1]
  [ 65  62  15]
  [ 76  77  20]]

 [[122 157  66]
  [120 155  58]
  [126 160  71]
  ...
  [ 22  16   3]
  [ 97 112  56]
  [141 161  87]]

 [[ 87 122  41]
  [ 88 122  39]
  [101 134  56]
  ...
  [ 34  36  10]
  [105 133  59]
  [138 173  79]]]
```

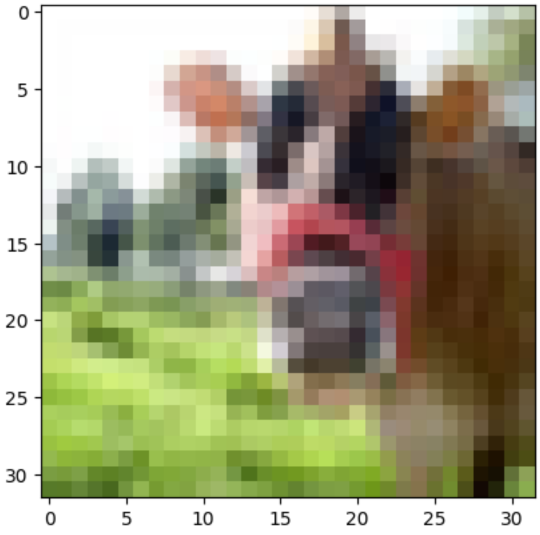
```
plt.figure(figsize=(10,10))
for i in range(50):
    plt.subplot(5,10,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    plt.xlabel(class_names[y_train[i][0]])
plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipython-input-1340203758.py in <cell line: 0>()
      6     plt.grid(False)
      7     plt.imshow(x_train[i])
----> 8     plt.xlabel(class_names[y_train[i][0]])
      9 plt.show()

IndexError: list index out of range
```

Next steps: Explain error

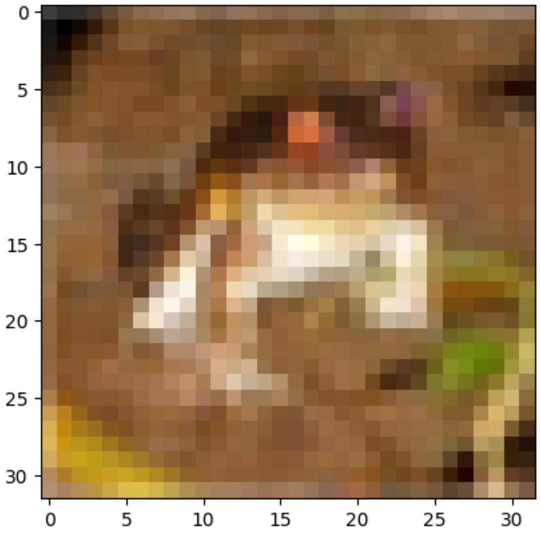
```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x7ef64f8aa0c0>
```



```
plt.imshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x7ef6c6da9700>
```

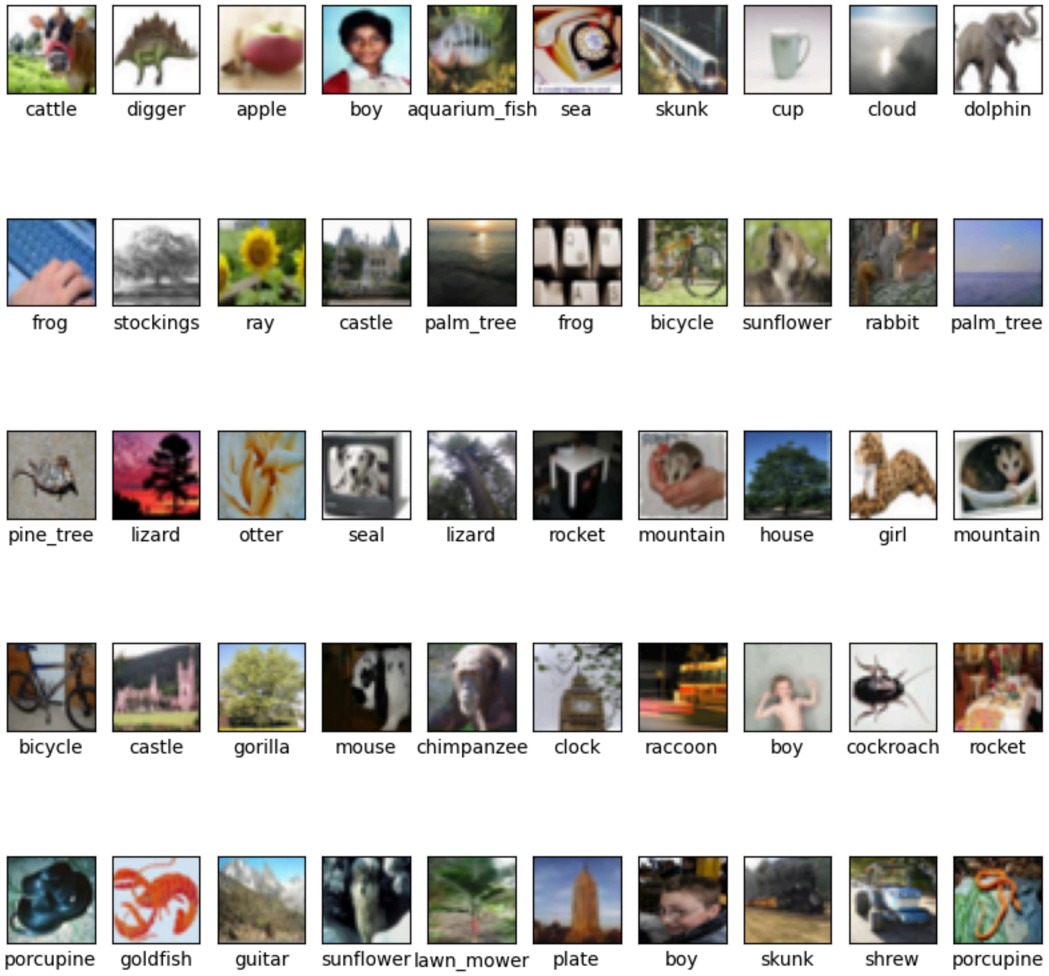


```
print([x_test[5]])
```

```
[array([[[179, 118,  83],
        [139,  96,  61],
        [ 77,  49,  26],
        ...,
        [ 87,  53,  46],
        [ 76,  47,  41],
        [ 77,  47,  41]],

       [[184, 130,  97],
        [133,  88,  53],
        [128,  89,  58],
        ...,
        [ 98,  61,  53],
        [ 91,  58,  51],
        [ 90,  57,  49]],

       [[180, 132, 100],
        [152, 104,  71],
        [176, 129,  92],
        ...,
        [101,  62,  53],
        [ 93,  56,  47],
        [ 95,  57,  49]],

       ...,

       [[142,  73,  61],
        [149,  84,  75],
        [144,  81,  73],
        ...,
        [119,  68,  56],
        [139,  87,  78],
        [159, 100,  89]],

       [[152,  83,  70],
        [166,  96,  81],
        [179, 106,  90],
        ...,
        [131,  77,  65],
        [144,  87,  77],
        [153,  90,  79]],

       [[159,  92,  79],
        [178, 107,  93],
        [183, 113,  95],
        ...,
        [150,  90,  76],
        [153,  91,  79],
        [152,  87,  73]]], dtype=uint8)]
```

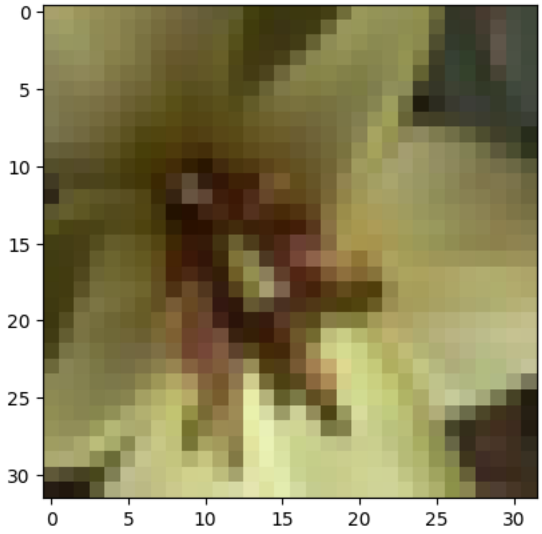
```
class_names = [
    'apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee', 'beetle', 'bicycle', 'bottle',
    'bowl', 'boy', 'bridge', 'bus', 'butterfly', 'camel', 'can', 'castle', 'caterpillar', 'cattle',
    'chair', 'chimpanzee', 'clock', 'cloud', 'cockroach', 'couch', 'crab', 'crocodile', 'cup', 'digger',
    'dinosaur', 'dolphin', 'elephant', 'ferret', 'finch', 'fish', 'flamingo', 'forest', 'fox', 'frog',
    'fruit_of_small_trees', 'giraffe', 'girl', 'globe', 'goat', 'goldfish', 'goose', 'gorilla', 'grass', 'guitar',
    'hamburger', 'hamster', 'house', 'kangaroo', 'keyboard', 'lamp', 'lawn_mower', 'leopard', 'lion', 'lizard',
    'lobster', 'man', 'maple_tree', 'motorcycle', 'mountain', 'mouse', 'mushroom', 'oak_tree', 'orange', 'orchid',
    'otter', 'palm_tree', 'pear', 'pickup_truck', 'pine_tree', 'plain', 'plate', 'poppy', 'porcupine', 'possum',
    'rabbit', 'raccoon', 'ray', 'road', 'rocket', 'rose', 'sea', 'seal', 'shark', 'shrew',
    'skunk', 'skyscraper', 'snail', 'snake', 'spider', 'squirrel', 'stockings', 'sunflower', 'sweet_pepper', 'table
    'tank', 'telephone', 'television', 'tiger', 'tractor', 'train', 'trout', 'tulip', 'turtle', 'wardrobe',
    'whale', 'willow_tree', 'wolf', 'woman', 'worm'
]
```

```
plt.figure(figsize=(10,10))
for i in range(50):
    plt.subplot(5,10,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    plt.xlabel(class_names[y_train[i][0]])
plt.show()
```



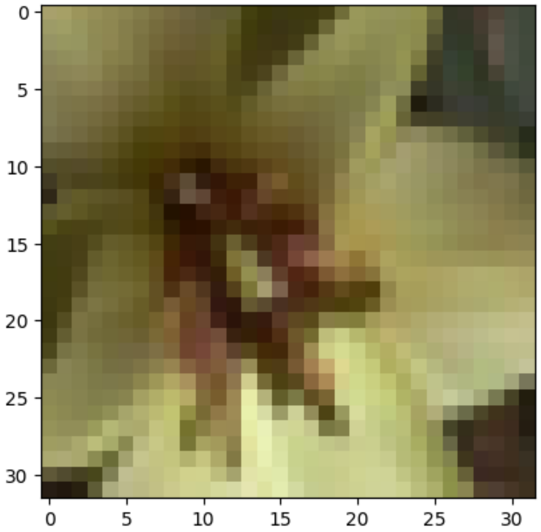
```
plt.imshow(x_test[5])
```

```
<matplotlib.image.AxesImage at 0x7ef67cb1af90>
```



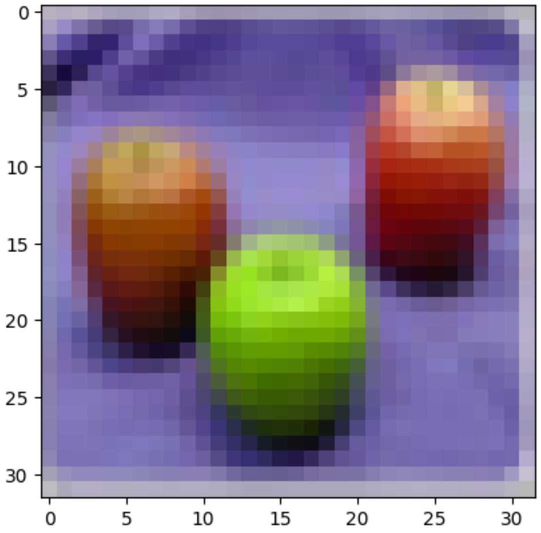
```
plt.imshow(x_test[5])
```

```
<matplotlib.image.AxesImage at 0x7ef67cbf2d80>
```



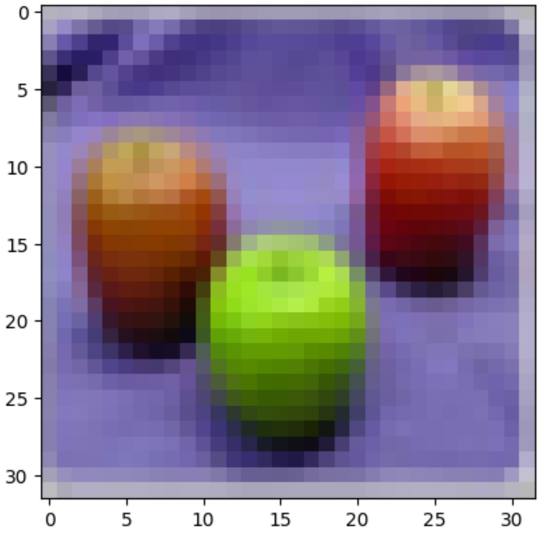
```
plt.imshow(x_test[9])
```

```
<matplotlib.image.AxesImage at 0x7ef67cb1a000>
```



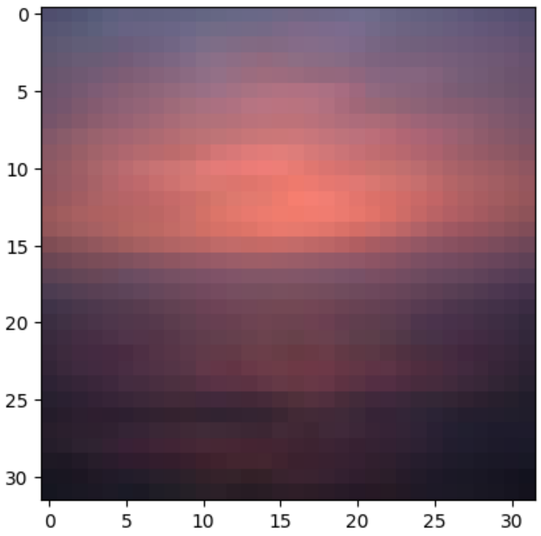
```
plt.imshow(x_test[9])
```

```
<matplotlib.image.AxesImage at 0x7ef67ccc59d0>
```



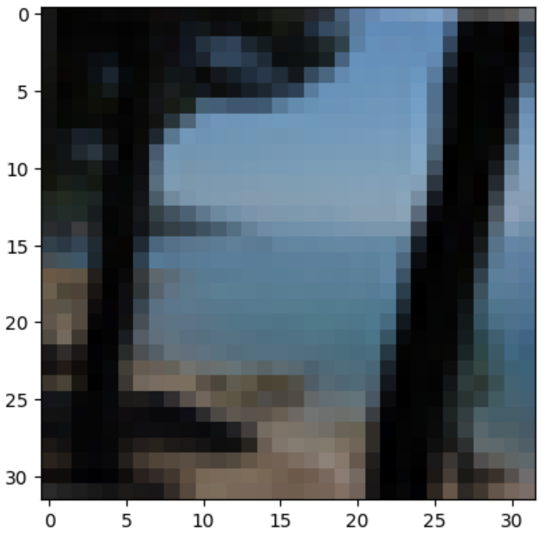
```
plt.imshow(x_test[4])
```

```
<matplotlib.image.AxesImage at 0x7ef67cd6c140>
```



```
plt.imshow(x_test[10])
```

```
<matplotlib.image.AxesImage at 0x7ef67cacb770>
```



```
#Scaling
x_train = x_train/255
x_test = x_test/255
```

```
print([x_train[0]])
```

```
[array([[[1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         [1.        , 1.        , 1.        ],
         ...,
         [0.76470588, 0.80392157, 0.75686275],
         [0.83137255, 0.87843137, 0.8       ],
         [0.71372549, 0.76078431, 0.65490196]],

        [[1.        , 1.        , 1.        ],
         [0.99607843, 0.99607843, 0.99607843],
         [0.99607843, 0.99607843, 0.99607843],
         ...,
         [0.66666667, 0.69019608, 0.58823529],
         [0.63137255, 0.65882353, 0.50980392],
         [0.57254902, 0.60392157, 0.44313725]],

        [[1.        , 1.        , 1.        ],
         [0.99607843, 0.99607843, 0.99607843],
         [1.        , 1.        , 1.        ],
         ...,
         [0.74117647, 0.78039216, 0.6627451 ],
         [0.65098039, 0.69803922, 0.50980392],
         [0.4745098 , 0.52156863, 0.34117647]],

        ...,

        [[0.58039216, 0.7254902 , 0.30980392],
         [0.55686275, 0.71372549, 0.22352941],
         [0.54901961, 0.70196078, 0.23529412],
         ...,
         [0.11764706, 0.06666667, 0.00392157],
         [0.25490196, 0.24313725, 0.05882353],
         [0.29803922, 0.30196078, 0.07843137]],

        [[0.47843137, 0.61568627, 0.25882353],
         [0.47058824, 0.60784314, 0.22745098],
         [0.49411765, 0.62745098, 0.27843137],
         ...,
         [0.08627451, 0.0627451 , 0.01176471],
         [0.38039216, 0.43921569, 0.21960784],
         [0.55294118, 0.63137255, 0.34117647]],

        [[0.34117647, 0.47843137, 0.16078431],
         [0.34509804, 0.47843137, 0.15294118],
         [0.39607843, 0.5254902 , 0.21960784],
         ...,
         [0.13333333, 0.14117647, 0.03921569],
         [0.41176471, 0.52156863, 0.23137255],
         [0.54117647, 0.67843137, 0.30980392]]]])
```

```
model=Sequential()
model.add(Flatten(input_shape=(32,32,3)))
model.add(Dense(128,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(100,activation='softmax'))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input
  super().__init__(**kwargs)
```

```
model.summary()
```

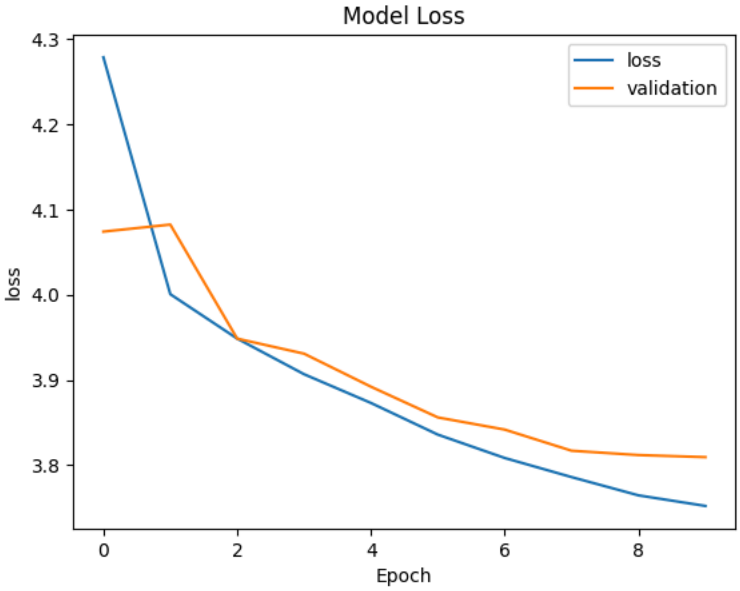
```
Model: "sequential_2"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ flatten_2 (Flatten)             │ (None, 3072)           │             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)
 Non-trainable params: 0 (0.00 B)
```

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

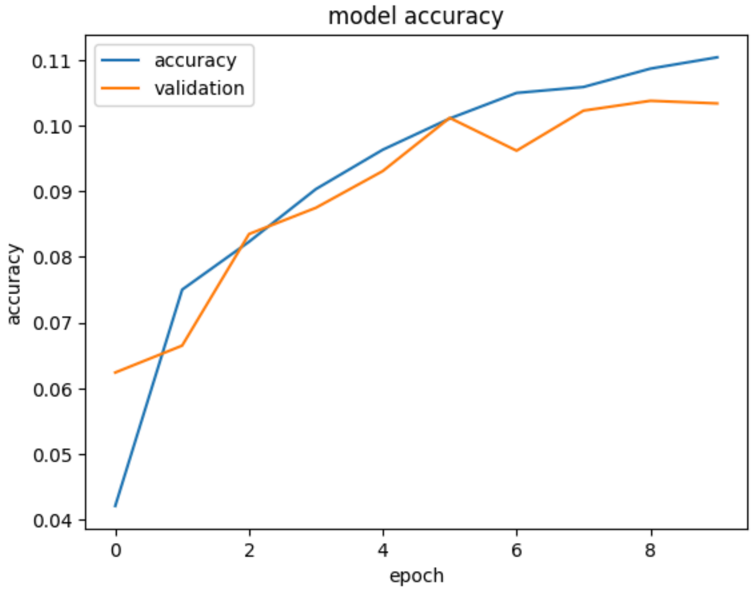
```
history=model.fit(x_train,y_train,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 13s 9ms/step - accuracy: 0.0263 - loss: 4.4546 - val_accuracy: 0.0624 - val_loss: 4.(
Epoch 2/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 13s 10ms/step - accuracy: 0.0723 - loss: 4.0203 - val_accuracy: 0.0665 - val_loss: 4.
Epoch 3/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 11s 9ms/step - accuracy: 0.0783 - loss: 3.9581 - val_accuracy: 0.0835 - val_loss: 3.9
Epoch 4/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 9ms/step - accuracy: 0.0892 - loss: 3.9104 - val_accuracy: 0.0875 - val_loss: 3.9
Epoch 5/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - accuracy: 0.0965 - loss: 3.8759 - val_accuracy: 0.0931 - val_loss: 3.8
Epoch 6/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 9ms/step - accuracy: 0.1017 - loss: 3.8430 - val_accuracy: 0.1012 - val_loss: 3.8
Epoch 7/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 9ms/step - accuracy: 0.1019 - loss: 3.8064 - val_accuracy: 0.0962 - val_loss: 3.8
Epoch 8/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 19s 8ms/step - accuracy: 0.1071 - loss: 3.7813 - val_accuracy: 0.1023 - val_loss: 3.8
Epoch 9/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 11s 9ms/step - accuracy: 0.1083 - loss: 3.7663 - val_accuracy: 0.1038 - val_loss: 3.8
Epoch 10/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 21s 9ms/step - accuracy: 0.1113 - loss: 3.7457 - val_accuracy: 0.1034 - val_loss: 3.8
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('loss')
plt.legend(['loss','validation'],loc='upper right')
plt.show()
```



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



```
model.evaluate(x_test, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.1025 - loss: 3.7934
[3.7997281551361084, 0.1062999963760376]
```

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```

```
y_pred.shape
```

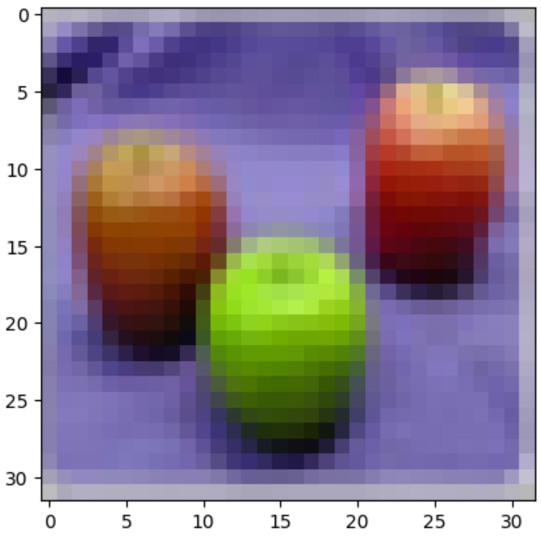
```
(10000, 100)
```

```
y_pred[0]
```

```
array([4.48935452e-06, 1.25140446e-04, 3.47745488e-03, 9.30038514e-04,
       8.13838269e-04, 2.61902865e-02, 2.22282833e-04, 2.35247120e-04,
       8.35625008e-02, 1.69467106e-02, 2.90874531e-03, 4.42481134e-03,
       5.16490787e-02, 2.94847619e-02, 1.22716383e-05, 5.96766360e-03,
       1.02104750e-02, 3.69996578e-02, 1.55959991e-04, 1.18478457e-03,
       2.20005983e-03, 8.75654281e-04, 1.52237983e-02, 1.08212288e-02,
       1.99181051e-03, 1.95544884e-02, 6.36901415e-04, 4.01641009e-03,
       1.42754009e-03, 2.59637157e-03, 1.54034421e-03, 2.23755580e-03,
       8.95250961e-03, 4.41505661e-04, 3.70181428e-04, 3.07251560e-03,
       6.02423039e-04, 2.58898064e-02, 1.27465953e-03, 2.73114219e-02,
       1.47255212e-02, 1.14810541e-02, 5.68932504e-04, 1.07716405e-05,
       6.32561510e-04, 6.59905258e-04, 4.36696364e-03, 3.90363421e-04,
       1.03832334e-02, 1.81706622e-02, 8.56069615e-04, 9.15679848e-05,
       7.80341579e-05, 2.94409062e-07, 3.04580363e-03, 4.65652067e-03,
       8.51555821e-03, 6.22133957e-04, 1.15133608e-02, 8.76763836e-03,
       6.07227609e-02, 3.37162893e-03, 1.16546209e-04, 1.15593371e-04,
       5.88174444e-04, 1.54149998e-03, 8.89140647e-04, 6.51518116e-04,
       2.80753057e-02, 1.22332081e-01, 3.99784913e-04, 4.21274640e-02,
       1.32761030e-02, 2.86203722e-04, 7.67981983e-05, 2.21064733e-03,
       4.43285778e-02, 4.68781102e-04, 1.31176971e-03, 8.43676645e-03,
       2.88410485e-03, 2.55322699e-02, 1.96867171e-04, 1.74713641e-05,
       8.11609160e-03, 5.34557663e-02, 3.04661551e-03, 1.40438657e-02,
       8.83462344e-05, 3.73718864e-03, 1.95257459e-02, 1.21326642e-02,
       7.10427485e-05, 2.29660748e-03, 3.74955311e-03, 7.64889689e-03,
       5.90666838e-04, 2.58995756e-03, 2.17733136e-03, 6.91574474e-04],
      dtype=float32)
```

```
plt.imshow(x_test[9])
predicted_class_index = y_pred[9].argmax()
print(f"Predicted class: {class_names[predicted_class_index]}")
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipython-input-3578696781.py in <cell line: 0>()
      1 plt.imshow(x_test[9])
      2 predicted_class_index = y_pred[9].argmax()
----> 3 print(f"Predicted class: {class_names[predicted_class_index]}")

IndexError: list index out of range
```



Next steps: Explain error

```
plt.imshow(x_test[0])
```

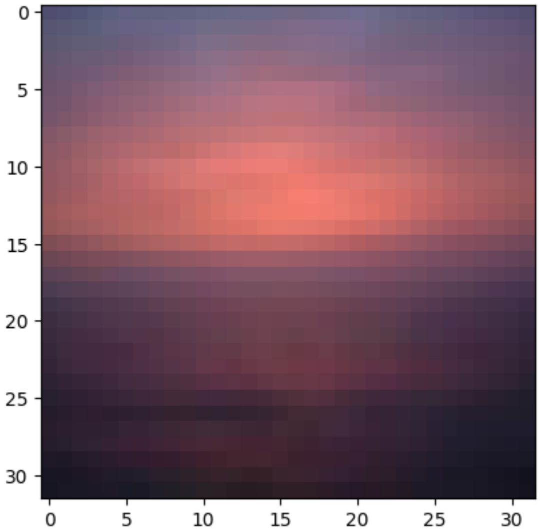
```
<matplotlib.image.AxesImage at 0x7ef679015d90>
```

```
print(y_pred[0].argmax())
```

```
69
```

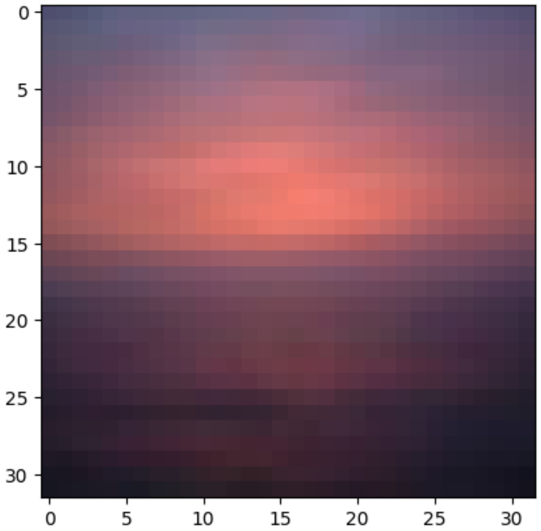
```
plt.imshow(x_test[4])
```

```
<matplotlib.image.AxesImage at 0x7ef678ea8f20>
```



```
plt.imshow(x_test[4])
predicted_class_index = y_pred[4].argmax()
print(f"Predicted class: {class_names[predicted_class_index]}")
```

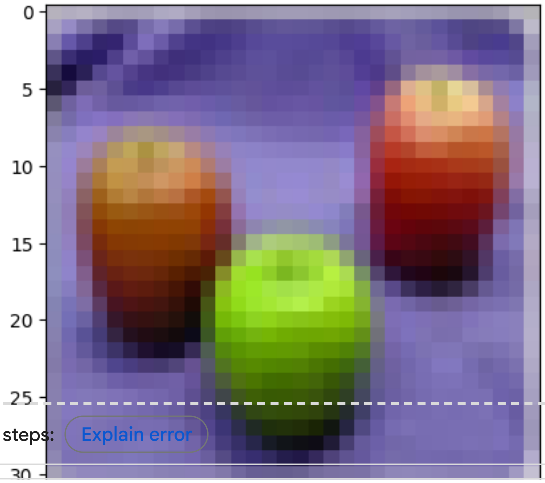
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipython-input-237728702.py in <cell line: 0>()
      1 plt.imshow(x_test[4])
      2 predicted_class_index = y_pred[4].argmax()
----> 3 print(f"Predicted class: {class_names[predicted_class_index]}")

IndexError: list index out of range
```



Next steps: Explain error

```
plt.imshow(x_test[9])
predicted_class_index = y_pred[9].argmax()
print(f"Predicted class: {class_names[predicted_class_index]}")
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipython-input-3578696781.py in <cell line: 0>()
      1 plt.imshow(x_test[9])
      2 predicted_class_index = y_pred[9].argmax()
----> 3 print(f"Predicted class: {class_names[predicted_class_index]}")

IndexError: list index out of range
```



Next steps: Explain error

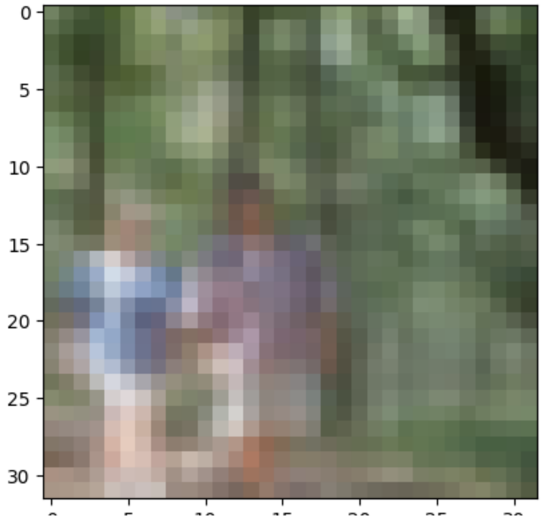
```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test, y_pred.argmax(axis=1))
```

```
0.1063
```

```
plt.imshow(x_test[1])
predicted_class_index = y_pred[1].argmax()
print(f"Predicted class: {class_names[predicted_class_index]}")
```

```
Predicted class: ship
```



```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

```
from tensorflow.keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

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

```
x_train.shape
```

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

```
x_test.shape
```

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

```
x_train[0]
```

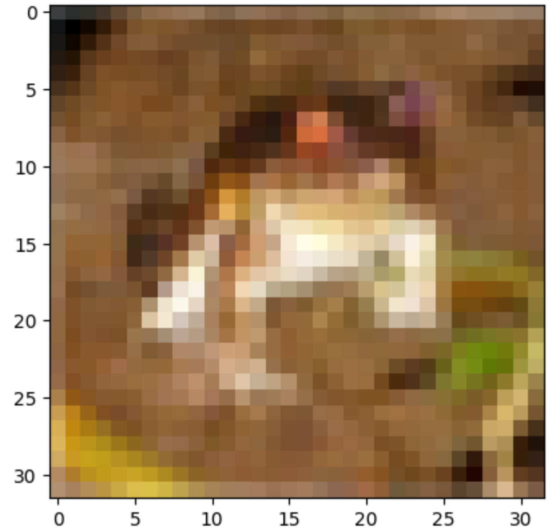
```
ndarray (32, 32, 3) show data
```

```
print([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)]
```

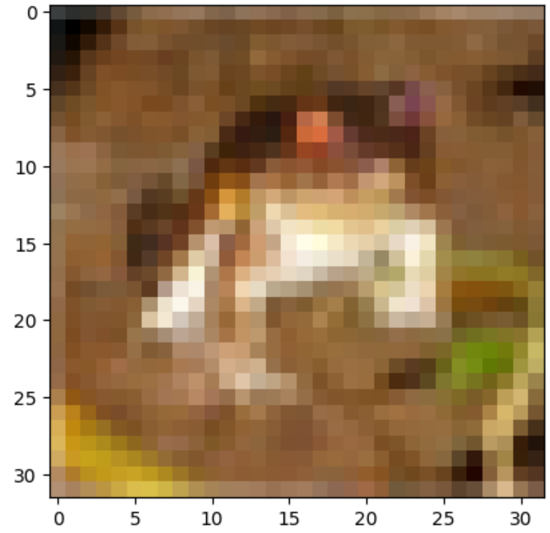
```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x7f1e82073d40>
```



```
plt.imshow(x_train[0], cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f1e822f9640>
```

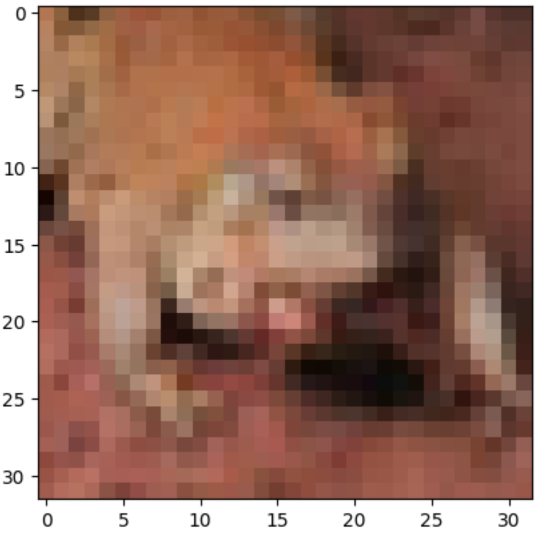


```
print([x_test[5]])
```

```
[array([[[179, 118,  83],
        [139,  96,  61],
        [ 77,  49,  26],
        ...,
        [ 87,  53,  46],
        [ 76,  47,  41],
        [ 77,  47,  41]],

       [[184, 130,  97],
        [133,  88,  53],
        [128,  89,  58],
        ...,
        [ 98,  61,  53],
        [ 91,  58,  51],
        [ 90,  57,  49]],

       [[180, 132, 100],
        [152, 104,  71],
        [176, 129,  92],
        ...,
        [101,  62,  53],
        [ 93,  56,  47],
        [ 95,  57,  49]],

       ...,

       [[142,  73,  61],
        [149,  84,  75],
        [144,  81,  73],
        ...,
        [119,  68,  56],
        [139,  87,  78],
        [159, 100,  89]],

       [[152,  83,  70],
        [166,  96,  81],
        [179, 106,  90],
        ...,
        [131,  77,  65],
        [144,  87,  77],
        [153,  90,  79]],

       [[159,  92,  79],
        [178, 107,  93],
        [183, 113,  95],
        ...,
        [150,  90,  76],
        [153,  91,  79],
        [152,  87,  73]]], dtype=uint8)]
```

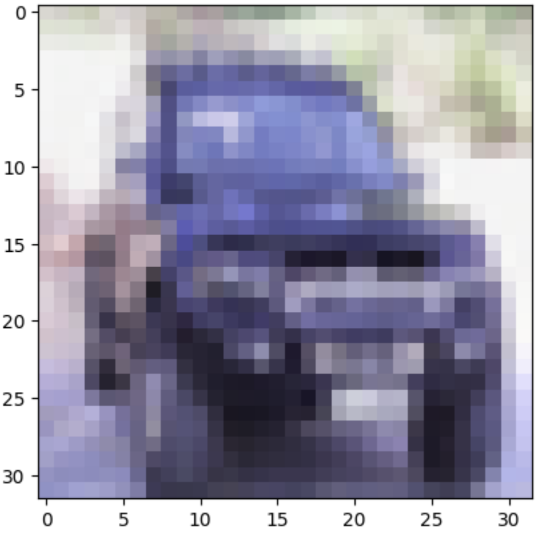
```
plt.imshow(x_test[5])
```

```
<matplotlib.image.AxesImage at 0x7f1e822cfc80>
```



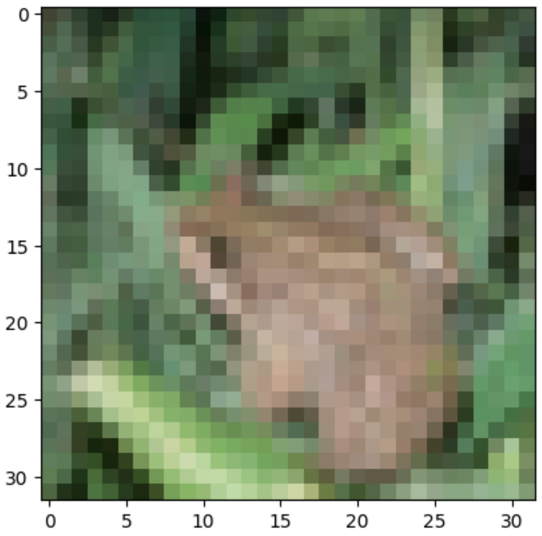
```
plt.imshow(x_test[9])
```

```
<matplotlib.image.AxesImage at 0x7f1e822b5580>
```



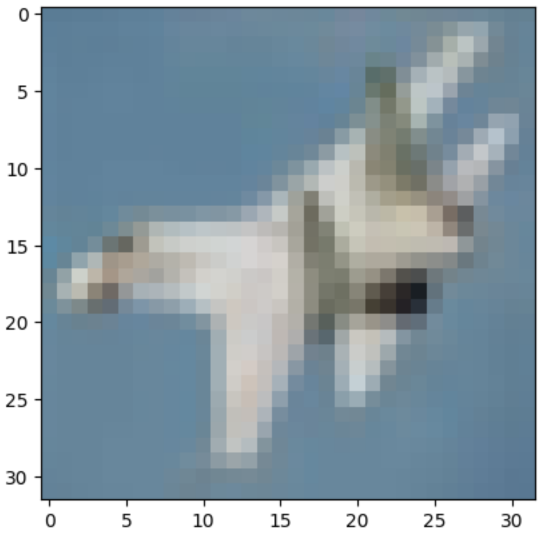
```
plt.imshow(x_test[4])
```

```
<matplotlib.image.AxesImage at 0x7f1e822b2300>
```



```
plt.imshow(x_test[10])
```

```
<matplotlib.image.AxesImage at 0x7f1e823c38c0>
```



```
# scaling
x_train = x_train/255
x_test = x_test/255
```

```
print([x_train[0]])
```

```
[array([[[0.23137255, 0.24313725, 0.24705882],
         [0.16862745, 0.18039216, 0.17647059],
         [0.19607843, 0.18823529, 0.16862745],
         ...,
         [0.61960784, 0.51764706, 0.42352941],
         [0.59607843, 0.49019608, 0.4       ],
         [0.58039216, 0.48627451, 0.40392157]],

        [[0.0627451 , 0.07843137, 0.07843137],
         [0.        , 0.        , 0.        ],
         [0.07058824, 0.03137255, 0.        ],
         ...,
         [0.48235294, 0.34509804, 0.21568627],
         [0.46666667, 0.3254902 , 0.19607843],
         [0.47843137, 0.34117647, 0.22352941]],

        [[0.09803922, 0.09411765, 0.08235294],
         [0.0627451 , 0.02745098, 0.        ],
         [0.19215686, 0.10588235, 0.03137255],
         ...,
         [0.4627451 , 0.32941176, 0.19607843],
         [0.47058824, 0.32941176, 0.19607843],
         [0.42745098, 0.28627451, 0.16470588]],

        ...,

        [[0.81568627, 0.66666667, 0.37647059],
         [0.78823529, 0.6       , 0.13333333],
         [0.77647059, 0.63137255, 0.10196078],
         ...,
         [0.62745098, 0.52156863, 0.2745098 ],
         [0.21960784, 0.12156863, 0.02745098],
         [0.20784314, 0.13333333, 0.07843137]],

        [[0.70588235, 0.54509804, 0.37647059],
         [0.67843137, 0.48235294, 0.16470588],
         [0.72941176, 0.56470588, 0.11764706],
         ...,
         [0.72156863, 0.58039216, 0.36862745],
         [0.38039216, 0.24313725, 0.13333333],
         [0.3254902 , 0.20784314, 0.13333333]],

        [[0.69411765, 0.56470588, 0.45490196],
         [0.65882353, 0.50588235, 0.36862745],
         [0.70196078, 0.55686275, 0.34117647],
         ...,
         [0.84705882, 0.72156863, 0.54901961],
         [0.59215686, 0.4627451 , 0.32941176],
         [0.48235294, 0.36078431, 0.28235294]]]])
```

```
model = Sequential()
model.add(Flatten(input_shape=(32, 32, 3)))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `inpu
  super().__init__(**kwargs)
```

```
model.summary()
```

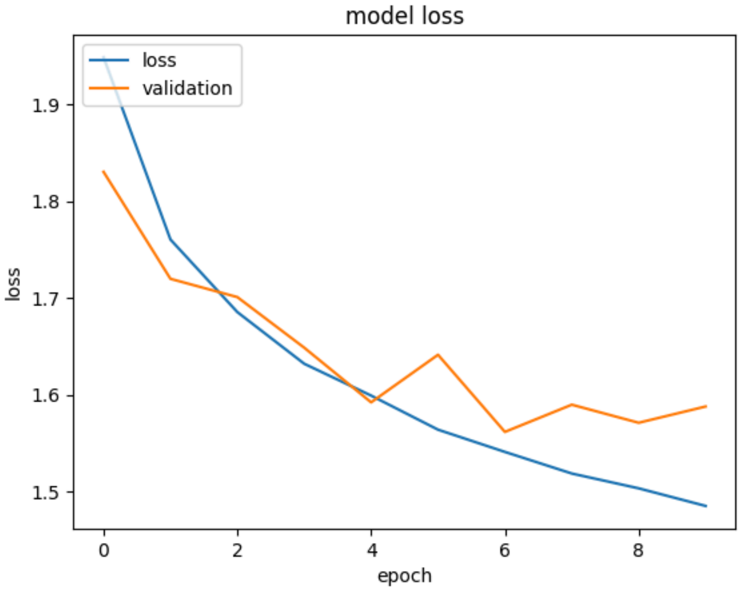
```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ flatten (Flatten)               │ (None, 3072)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │       393,344 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 64)             │         8,256 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 32)             │         2,080 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                 │ (None, 10)             │           330 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 404,010 (1.54 MB)
 Trainable params: 404,010 (1.54 MB)
 Non-trainable params: 0 (0.00 B)
```

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

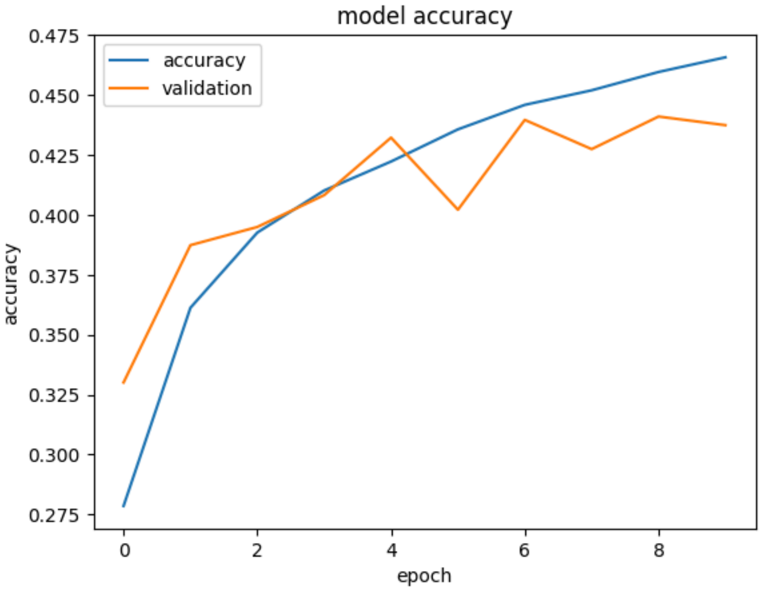
```
history=model.fit(x_train, y_train, epochs=10 ,validation_split=0.2)
```

```
Epoch 1/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 7ms/step - accuracy: 0.2236 - loss: 2.0703 - val_accuracy: 0.3301 - val_loss: 1.8
Epoch 2/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 7ms/step - accuracy: 0.3570 - loss: 1.7740 - val_accuracy: 0.3874 - val_loss: 1.7
Epoch 3/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 7ms/step - accuracy: 0.3863 - loss: 1.6988 - val_accuracy: 0.3950 - val_loss: 1.7
Epoch 4/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.4095 - loss: 1.6291 - val_accuracy: 0.4083 - val_loss: 1.64
Epoch 5/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 8s 6ms/step - accuracy: 0.4242 - loss: 1.5928 - val_accuracy: 0.4323 - val_loss: 1.59
Epoch 6/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.4365 - loss: 1.5585 - val_accuracy: 0.4022 - val_loss: 1.64
Epoch 7/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 8s 6ms/step - accuracy: 0.4455 - loss: 1.5432 - val_accuracy: 0.4397 - val_loss: 1.56
Epoch 8/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.4536 - loss: 1.5192 - val_accuracy: 0.4275 - val_loss: 1.5
Epoch 9/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.4633 - loss: 1.5001 - val_accuracy: 0.4411 - val_loss: 1.5
Epoch 10/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.4679 - loss: 1.4847 - val_accuracy: 0.4375 - val_loss: 1.58
```

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



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



```
model.evaluate(x_test, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.4420 - loss: 1.5557
[1.5610451698303223, 0.4465999901294708]
```

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

```
y_pred.shape
```

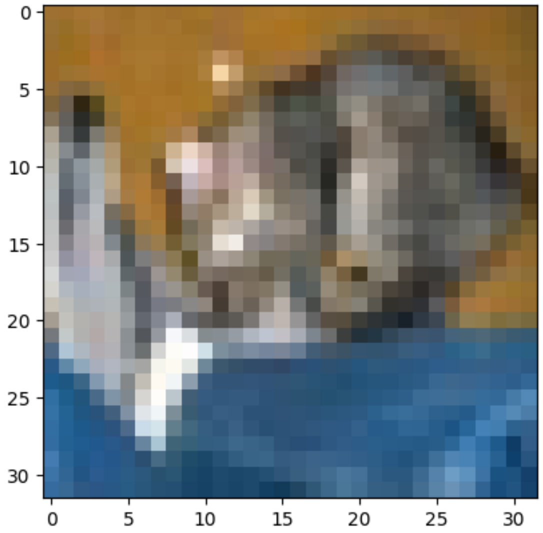
```
(10000, 10)
```

```
y_pred[0]
```

```
array([0.02442311, 0.11145618, 0.06991897, 0.37400377, 0.04220996,
       0.14988008, 0.0752502 , 0.01981828, 0.09571822, 0.03732129],
      dtype=float32)
```

```
plt.imshow(x_test[0])
```

```
<matplotlib.image.AxesImage at 0x7f1e16f79f70>
```



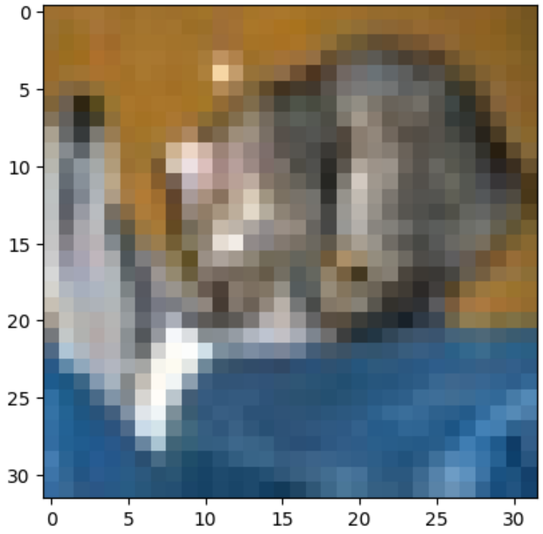
```
print(y_pred[0].argmax())
```

```
3
```

```
Start coding or generate with AI.
```

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
plt.imshow(x_test[0])
predicted_class_index = y_pred[0].argmax()
print(f"Predicted class: {class_names[predicted_class_index]}")
```

```
Predicted class: cat
```



```
y_pred[4]
```

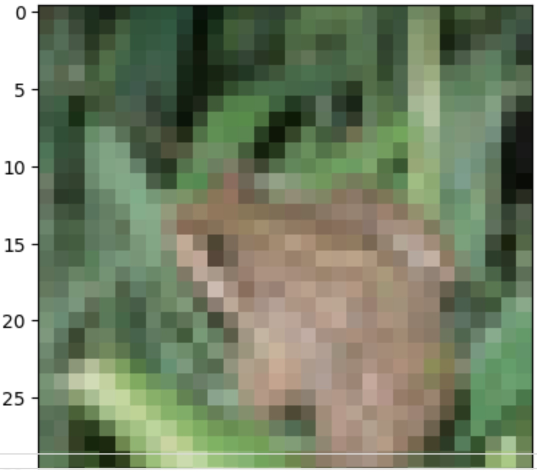
```
array([0.00996003, 0.0018735 , 0.11183698, 0.02911268, 0.6914521 ,
       0.02661325, 0.09309293, 0.02653859, 0.00852393, 0.00099599],
      dtype=float32)
```

```
print(y_pred[4].argmax())
```

```
4
```

```
plt.imshow(x_test[4])
predicted_class_index = y_pred[4].argmax()
print(f"Predicted class: {class_names[predicted_class_index]}")
```

```
Predicted class: deer
```



```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test, y_pred.argmax(axis=1))
```

```
0.4466
```

```
Start coding or generate with AI.
```