Task 4 Machine Learning Internship:
Haris Khan

[63]:
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
```

[64]:
```
(X_train, y_train), (X_test,y_test) = datasets.cifar10.load_data()
X_train.shape
```

[64]:
```
(50000, 32, 32, 3)
```

[65]:
```
X_test.shape
```

[65]:
```
(10000, 32, 32, 3)
```

[66]:
```
y_train.shape
```

[66]:
```
(50000, 1)
```

[67]:
```
y_train[:5]
```

[67]:
```
array([[6],
       [9],
       [9],
       [4],
       [1]], dtype=uint8)
```

[68]:
```
y_train = y_train.reshape(-1,)
y_train[:5]
```

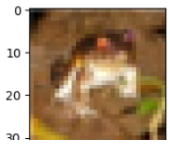
[68]:
```
array([6, 9, 9, 4, 1], dtype=uint8)
```

[69]:
```
y_test = y_test.reshape(-1,)
```

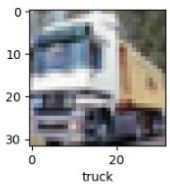
[70]:
```
classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

[71]:
```
def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

[72]:
```
plot_sample(X_train, y_train, 0)
```



[73]:
```
plot_sample(X_train, y_train, 1)
```



[74]:
```
X_train = X_train / 255.0
X_test = X_test / 255.0
```

[75]:
```
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)
```

```
/home/haris-khan/jupyter-env/lib/python3.12/site-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Epoch 1/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 60s 38ms/step - accuracy: 0.3028 - loss: 1.9363
Epoch 2/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 59s 38ms/step - accuracy: 0.4193 - loss: 1.6480
Epoch 3/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 60s 39ms/step - accuracy: 0.4546 - loss: 1.5531
Epoch 4/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 63s 40ms/step - accuracy: 0.4702 - loss: 1.4992
Epoch 5/5
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 62s 40ms/step - accuracy: 0.4926 - loss: 1.4468
```

[75]:
```
<keras.src.callbacks.history.History at 0x7c0ca3755460>
```

[41]:
```
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]
```

[41]:
```
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 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.57      0.54      1000
           1       0.74      0.41      0.53      1000
           2       0.47      0.16      0.24      1000
           3       0.32      0.31      0.32      1000
           4       0.63      0.12      0.21      1000
           5       0.32      0.48      0.38      1000
           6       0.41      0.71      0.52      1000
           7       0.48      0.63      0.54      1000
           8       0.55      0.69      0.62      1000
           9       0.56      0.58      0.57      1000

    accuracy                           0.47     10000
   macro avg       0.50      0.47      0.45     10000
weighted avg       0.50      0.47      0.45     10000
```

[42]:
```
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')
])
```

```
/home/haris-khan/jupyter-env/lib/python3.12/site-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)
```

[43]:
```
cnn.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])
```

[44]:
```
cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 16ms/step - accuracy: 0.3968 - loss: 1.6608
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 17ms/step - accuracy: 0.6104 - loss: 1.1177
Epoch 3/10
```

[44]:
```
cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 16ms/step - accuracy: 0.3968 - loss: 1.6608
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 17ms/step - accuracy: 0.6104 - loss: 1.1177
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 16ms/step - accuracy: 0.6637 - loss: 0.9710
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 16ms/step - accuracy: 0.6951 - loss: 0.8733
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 16ms/step - accuracy: 0.7276 - loss: 0.7950
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 16ms/step - accuracy: 0.7400 - loss: 0.7422
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 17ms/step - accuracy: 0.7629 - loss: 0.6793
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 17ms/step - accuracy: 0.7811 - loss: 0.6326
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 17ms/step - accuracy: 0.7960 - loss: 0.5829
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 26s 17ms/step - accuracy: 0.8096 - loss: 0.5477
```

[44]:
```
<keras.src.callbacks.history.History at 0x7c0d09f625d0>
```

[45]:
```
y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```

[45]:
```
array([[2.5352151e-06, 2.6942054e-05, 1.7831140e-04, 9.6407247e-01,
        1.3295193e-04, 2.1179426e-02, 1.1658215e-02, 9.5442022e-07,
        2.7371121e-03, 1.1200614e-05],
       [3.0432458e-03, 2.3638444e-02, 1.5797217e-04, 8.7096168e-06,
        1.4406257e-07, 6.5121561e-08, 3.8977470e-07, 1.1124809e-07,
        9.6196413e-01, 1.1186650e-02],
       [2.2752885e-02, 9.5861517e-02, 1.6352712e-04, 1.3839080e-03,
        4.7590127e-04, 2.4885873e-04, 6.0887010e-05, 1.1321453e-03,
        7.9294497e-01, 8.4975407e-02],
       [7.8561568e-01, 1.1721703e-02, 1.6631883e-01, 1.0808820e-03,
        9.8324716e-03, 1.2070191e-04, 7.9765778e-05, 2.2857585e-04,
        2.4680052e-02, 3.2115870e-04],
       [1.6112288e-07, 2.5472340e-05, 9.5093315e-03, 1.5235729e-02,
        1.6124251e-01, 1.5780315e-03, 8.1240332e-01, 1.0699318e-06,
        4.0293176e-06, 4.0363693e-07]], dtype=float32)
```

[46]:
```
y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

[46]:
```
[3, 8, 8, 0, 6]
```

[47]:
```
y_test[:5]
```

[47]:
```
array([3, 8, 8, 0, 6], dtype=uint8)
```

[48]:
```
plot_sample(X_test, y_test,3)
```

[45]:
```
array([[2.5552151e-06, 2.6942054e-05, 1.7831148e-04, 9.6407247e-01,
        1.3295193e-04, 2.1179426e-02, 1.1658215e-02, 9.5442022e-07,
        2.7371121e-03, 1.1200614e-05],
       [3.0432458e-03, 2.3638444e-02, 1.5797217e-04, 8.7096168e-06,
        1.4406257e-07, 6.5121561e-08, 3.8977470e-07, 1.1124809e-07,
        9.6196413e-01, 1.1186650e-02],
       [2.2752885e-02, 9.5861517e-02, 1.6352712e-04, 1.3839080e-03,
        4.7590127e-04, 2.4885873e-04, 6.0887010e-05, 1.1321453e-03,
        7.9294497e-01, 8.4975407e-02],
       [7.8561568e-01, 1.1721703e-02, 1.6631883e-01, 1.0808820e-03,
        9.8324716e-03, 1.2070191e-04, 7.9765778e-05, 2.2857585e-04,
        2.4680052e-02, 3.2115870e-04],
       [1.6112288e-07, 2.5472340e-05, 9.5093315e-03, 1.5235729e-02,
        1.6124251e-01, 1.5780315e-03, 8.1240332e-01, 1.0699318e-06,
        4.0293176e-06, 4.0363693e-07]], dtype=float32)
```

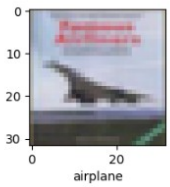
[46]:
```
y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

[46]:
```
[3, 8, 8, 0, 6]
```

[47]:
```
y_test[:5]
```

[47]:
```
array([3, 8, 8, 0, 6], dtype=uint8)
```

[48]:
```
plot_sample(X_test, y_test,3)
```



[49]:
```
classes[y_classes[3]]
```

[49]:
```
'airplane'
```

[50]:
```
classes[y_classes[3]]
```

[50]:
```
'airplane'
```

[ ]: