Image_Classification

May 26, 2024

[1]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, models
import tensorflow.keras.layers as tfl
from tensorflow.keras.utils import to_categorical
from tensorflow.math import confusion_matrix
from tensorflow.keras.utils import to_categorical
```

[2]:

```
(train_X, train_Y), (test_X, test_Y) = datasets.cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
```

[3]:

```
print(train_X.shape)
print(test_X.shape)
print(train_Y.shape)
print(test_Y.shape)
```

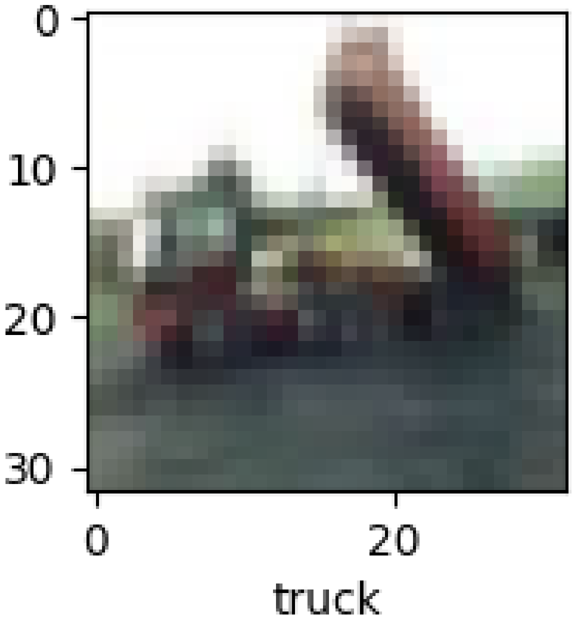
```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
(50000, 1)
(10000, 1)
```

[4]:

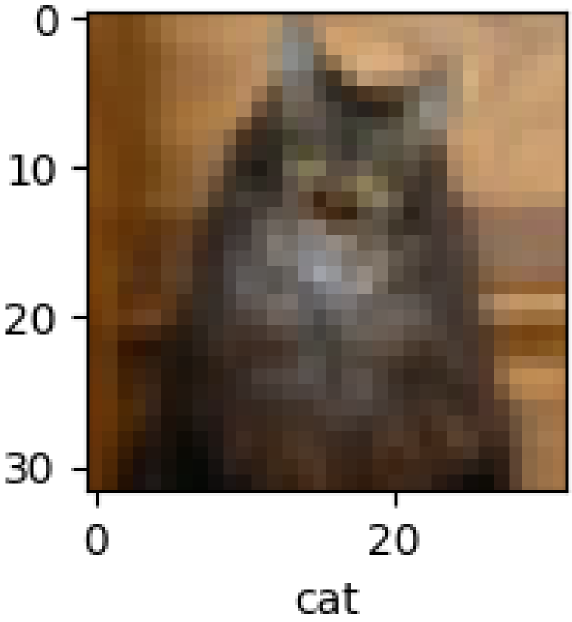
```
classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog",␣
 ↪"horse", "ship", "truck"]

def plot_image(X,Y,index):
  plt.figure(figsize=(15,2))
  plt.imshow(X[index])
  plt.xlabel(classes[Y[index][0]])
```

[5]:

```
plot_image(train_X, train_Y, 2)
```



[6]:
```
plot_image(train_X, train_Y, 26)
```



[7]:

```
train_X = train_X/255
test_X = test_X/255

train_Y = to_categorical(train_Y)
test_Y = to_categorical(test_Y)
```

[8]:

```
ann = models.Sequential([
    tfl.Flatten(input_shape=(32,32,3)),
    tfl.Dense(3000, activation='relu'),
    tfl.Dense(1000, activation='relu'),
    tfl.Dense(10,activation='sigmoid')
])
```

[9]:

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

[10]:

```
ann.fit(train_X, train_Y, epochs=5)
```

```
Epoch 1/5
1563/1563 [==============================] - 11s 6ms/step - loss: 1.8758 -
accuracy: 0.3295
Epoch 2/5
1563/1563 [==============================] - 9s 6ms/step - loss: 1.6785 -
accuracy: 0.3971
Epoch 3/5
1563/1563 [==============================] - 10s 7ms/step - loss: 1.5995 -
accuracy: 0.4254
Epoch 4/5
1563/1563 [==============================] - 10s 7ms/step - loss: 1.5456 -
accuracy: 0.4455
Epoch 5/5
1563/1563 [==============================] - 11s 7ms/step - loss: 1.5117 -
accuracy: 0.4599
```

[10]:

```
<keras.src.callbacks.History at 0x78955bcaa1d0>
```

[11]:

```
ann.evaluate(test_X,test_Y)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 1.5255 -
accuracy: 0.4623
```

[11]:

```
[1.5255182981491089, 0.46230000257492065]
```

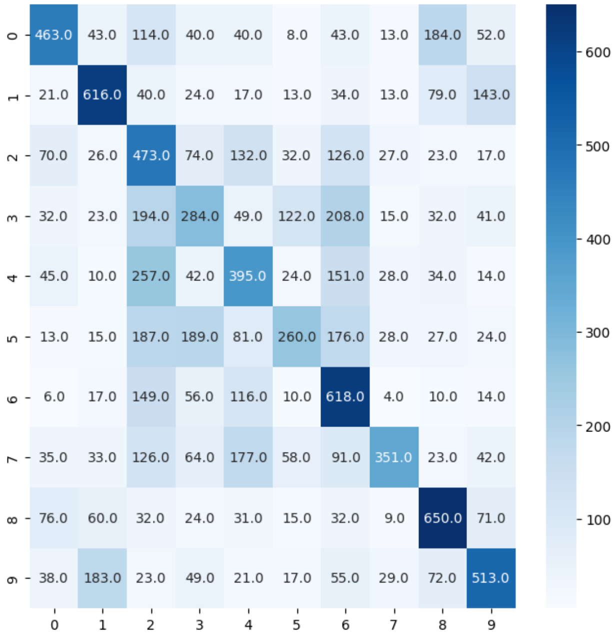
[12]:
```
pred_test_Y = ann.predict(test_X)
```

```
313/313 [==============================] - 1s 2ms/step
```

[13]:
```
conf_mat = confusion_matrix(test_Y.argmax(axis=1), pred_test_Y.argmax(axis=1))
```

[14]:
```
import seaborn as sns
plt.figure(figsize=(8,8))
sns.heatmap(conf_mat, annot=True, fmt='.1f', cmap='Blues')
```

[14]: <Axes: >



	0	1	2	3	4	5	6	7	8	9
0	463.0	43.0	114.0	40.0	40.0	8.0	43.0	13.0	184.0	52.0
1	21.0	616.0	40.0	24.0	17.0	13.0	34.0	13.0	79.0	143.0
2	70.0	26.0	473.0	74.0	132.0	32.0	126.0	27.0	23.0	17.0
3	32.0	23.0	194.0	284.0	49.0	122.0	208.0	15.0	32.0	41.0
4	45.0	10.0	257.0	42.0	395.0	24.0	151.0	28.0	34.0	14.0
5	13.0	15.0	187.0	189.0	81.0	260.0	176.0	28.0	27.0	24.0
6	6.0	17.0	149.0	56.0	116.0	10.0	618.0	4.0	10.0	14.0
7	35.0	33.0	126.0	64.0	177.0	58.0	91.0	351.0	23.0	42.0
8	76.0	60.0	32.0	24.0	31.0	15.0	32.0	9.0	650.0	71.0
9	38.0	183.0	23.0	49.0	21.0	17.0	55.0	29.0	72.0	513.0

[15]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[16]:
```
print(classification_report(test_Y.argmax(axis=1), pred_test_Y.argmax(axis=1) ))
```

```
              precision    recall  f1-score   support

           0       0.58      0.46      0.51      1000
           1       0.60      0.62      0.61      1000
           2       0.30      0.47      0.36      1000
           3       0.34      0.28      0.31      1000
           4       0.37      0.40      0.38      1000
           5       0.47      0.26      0.33      1000
           6       0.40      0.62      0.49      1000
           7       0.68      0.35      0.46      1000
           8       0.57      0.65      0.61      1000
           9       0.55      0.51      0.53      1000

    accuracy                           0.46     10000
   macro avg       0.49      0.46      0.46     10000
weighted avg       0.49      0.46      0.46     10000
```

[17]:
```
cnn = models.Sequential([

        # cnn
        tfl.Conv2D(filters=32, kernel_size=(3,3), activation='relu',␣
 ↪input_shape=(32,32,3)),
        tfl.MaxPooling2D((2,2)),

        tfl.Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
        tfl.MaxPooling2D((2,2)),

        tfl.Conv2D(filters=128, kernel_size=(3,3), activation='relu'),
        tfl.MaxPooling2D((2,2)),

        # Dense
        tfl.Flatten(),
        tfl.Dense(128, activation='relu'),
        tfl.Dense(10, activation='softmax')

    ])
```

[18]:
```
cnn.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy'])
```

[19]:
```
cnn.fit(train_X, train_Y, epochs=5)
```

```
Epoch 1/5
1563/1563 [==============================] - 14s 6ms/step - loss: 1.4877 -
accuracy: 0.4564
Epoch 2/5
1563/1563 [==============================] - 11s 7ms/step - loss: 1.0982 -
accuracy: 0.6124
Epoch 3/5
1563/1563 [==============================] - 12s 7ms/step - loss: 0.9406 -
accuracy: 0.6711
Epoch 4/5
1563/1563 [==============================] - 13s 8ms/step - loss: 0.8412 -
accuracy: 0.7042
Epoch 5/5
```

```
1563/1563 [==============================] - 11s 7ms/step - loss: 0.7643 -
accuracy: 0.7322
```

[19]:
```
<keras.src.callbacks.History at 0x78955c694760>
```

[20]:
```
cnn.evaluate(test_X, test_Y)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.9857 -
accuracy: 0.6742
```

[20]:
```
[0.9857380986213684, 0.6741999983787537]
```

[21]:
```
pred_test_Y = cnn.predict(test_X)
```

```
313/313 [==============================] - 1s 2ms/step
```

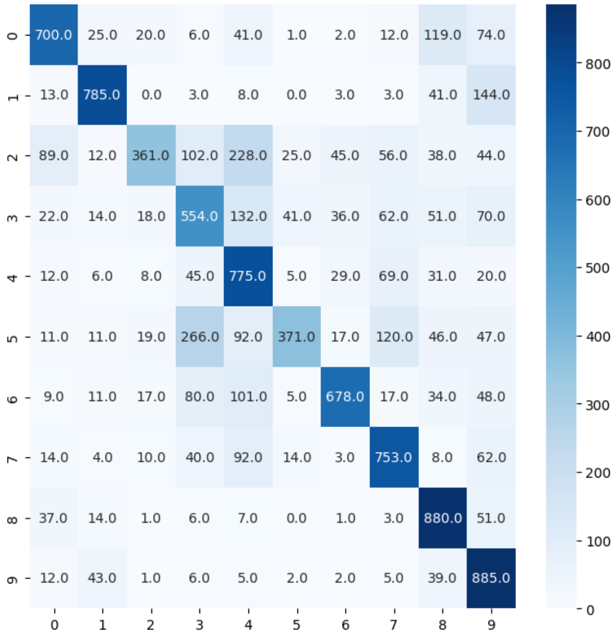
[22]:
```
conf_matrix = confusion_matrix(test_Y.argmax(axis=1), pred_test_Y.argmax(axis=1))
```

[23]:
```
print(conf_matrix)
```

```
[[700  25  20   6  41   1   2  12 119  74]
 [ 13 785   0   3   8   0   3   3  41 144]
 [ 89  12 361 102 228  25  45  56  38  44]
 [ 22  14  18 554 132  41  36  62  51  70]
 [ 12   6   8  45 775   5  29  69  31  20]
 [ 11  11  19 266  92 371  17 120  46  47]
 [  9  11  17  80 101   5 678  17  34  48]
 [ 14   4  10  40  92  14   3 753   8  62]
 [ 37  14   1   6   7   0   1   3 880  51]
 [ 12  43   1   6   5   2   2   5  39 885]]
```

[24]:
```
plt.figure(figsize=(8,8))
sns.heatmap(conf_matrix, annot=True, fmt='.1f', cmap='Blues')
```

[24]:
```
<Axes: >
```



[25]:
```
print(classification_report(test_Y.argmax(axis=1), pred_test_Y.argmax(axis=1)))
```

```
              precision    recall  f1-score   support

           0       0.76      0.70      0.73      1000
           1       0.85      0.79      0.82      1000
           2       0.79      0.36      0.50      1000
           3       0.50      0.55      0.53      1000
           4       0.52      0.78      0.62      1000
           5       0.80      0.37      0.51      1000
           6       0.83      0.68      0.75      1000
           7       0.68      0.75      0.72      1000
           8       0.68      0.88      0.77      1000
           9       0.61      0.89      0.72      1000

    accuracy                           0.67     10000
   macro avg       0.70      0.67      0.67     10000
weighted avg       0.70      0.67      0.67     10000
```