DEEP LEARNING CNN FOR IMG

May 11, 2023

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

1 Load the dataset

```
[14]: (x_train,y_train),(x_test,y_test)= datasets.cifar10.load_data()
      x_train.shape
```

```
[14]: (50000, 32, 32, 3)
```

```
[15]: x_test.shape
```

```
[15]: (10000, 32, 32, 3)
```

```
[16]: y_train.shape
```

```
[16]: (50000, 1)
```

```
[17]: y_train[:5]
```

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

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

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

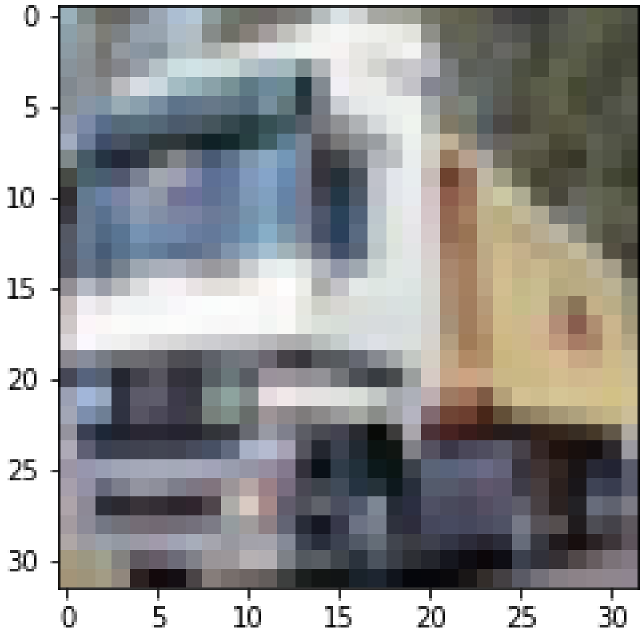
```
[20]: y_test=y_test.reshape(-1)
      y_test[:5]
```

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

[21]:
```
classes =␣
  ↪["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

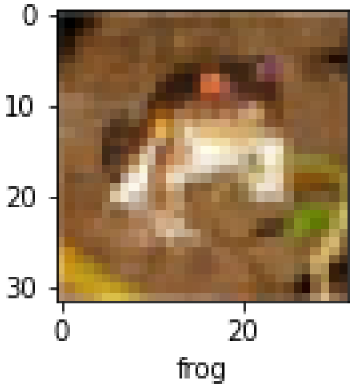
[26]:
```
plt.imshow(x_train[1])
```

[26]:
```
<matplotlib.image.AxesImage at 0x21ff837d580>
```



[36]:
```
def plot_sample(x, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(x[index])
    plt.xlabel(classes[y[index]])
```

[38]:
```
plot_sample(x_train, y_train,0)
```



Normalizing the training data

[41]:
```
x_train = x_train / 255.0
x_test = x_test / 255.0
```

Build simple artificial neural network for image classification

[46]:
```
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)
```

```
Epoch 1/5
1563/1563 [==============================] - 108s 68ms/step - loss: 1.8119 -
accuracy: 0.3535
Epoch 2/5
1563/1563 [==============================] - 116s 74ms/step - loss: 1.6212 -
accuracy: 0.4289
Epoch 3/5
1563/1563 [==============================] - 112s 72ms/step - loss: 1.5388 -
accuracy: 0.4604
Epoch 4/5
1563/1563 [==============================] - 114s 73ms/step - loss: 1.4784 -
accuracy: 0.4816
Epoch 5/5
1563/1563 [==============================] - 113s 72ms/step - loss: 1.4305 -
accuracy: 0.4958
```

[46]:
```
<keras.callbacks.History at 0x21ff9e58d00>
```

[ ]:
```
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
y_pred=ann.pred(x_train)
y_pred_classes=[np.argmax(i) for i in y_pred]
print('classification report:\n',classification_report(y_test,y_pred_classes))
```

Now let us build a convolutional neural network to train our images

[50]:
```
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=32,kernel_size=(3,3),activation='relu',input_shape=(32, 32,␣
 ↪3)),
    layers.MaxPooling2D(2,2),

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

])
```

[51]:
```
cnn.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[52]:
```
cnn.fit(x_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 [==============================] - 55s 34ms/step - loss: 1.4824 -
accuracy: 0.4654
Epoch 2/10
1563/1563 [==============================] - 46s 30ms/step - loss: 1.1434 -
accuracy: 0.5986
Epoch 3/10
1563/1563 [==============================] - 46s 30ms/step - loss: 1.0173 -
accuracy: 0.6467
Epoch 4/10
1563/1563 [==============================] - 47s 30ms/step - loss: 0.9416 -
accuracy: 0.6740
Epoch 5/10
1563/1563 [==============================] - 46s 30ms/step - loss: 0.8898 -
accuracy: 0.6895
Epoch 6/10
1563/1563 [==============================] - 47s 30ms/step - loss: 0.8412 -
accuracy: 0.7092
Epoch 7/10
1563/1563 [==============================] - 47s 30ms/step - loss: 0.8042 -
accuracy: 0.7204
Epoch 8/10
1563/1563 [==============================] - 47s 30ms/step - loss: 0.7729 -
accuracy: 0.7310
Epoch 9/10
1563/1563 [==============================] - 47s 30ms/step - loss: 0.7413 -
```

```
accuracy: 0.7417
Epoch 10/10
1563/1563 [==============================] - 46s 29ms/step - loss: 0.7185 -
accuracy: 0.7508
```

```
[52]: <keras.callbacks.History at 0x21ff44dc970>
```

```
[53]: cnn.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 4s 10ms/step - loss: 0.9174 -
accuracy: 0.6870
```

```
[53]: [0.9173777103424072, 0.6869999766349792]
```

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

```
313/313 [==============================] - 3s 10ms/step
```

```
[54]: array([[1.9725633e-03, 1.5894047e-03, 3.4172602e-02, 6.2655628e-01,
              1.1393116e-03, 2.4108025e-01, 5.3924888e-02, 7.4508460e-04,
              3.8541365e-02, 2.7816434e-04],
             [7.5108828e-03, 6.8859190e-01, 8.8441222e-05, 3.8812198e-07,
              1.3799688e-06, 3.3285879e-08, 2.4247137e-07, 2.9485159e-06,
              2.9701161e-01, 6.7921886e-03],
             [2.8895989e-02, 2.9311344e-01, 3.5146768e-03, 3.6700168e-03,
              7.7218062e-04, 8.3693897e-04, 3.4413037e-03, 1.4111316e-03,
              6.3737273e-01, 2.6971566e-02],
             [9.6753818e-01, 5.6178086e-03, 1.9537913e-02, 7.7568390e-04,
              5.4022152e-04, 4.8479553e-05, 1.6037687e-03, 1.6161461e-03,
              2.6224009e-03, 9.9434881e-05],
             [3.0197575e-06, 8.8355891e-06, 3.1184141e-02, 3.6131172e-03,
              6.2262379e-02, 4.2312237e-04, 9.0248674e-01, 3.6710088e-07,
              1.8193814e-05, 2.3719113e-07]], dtype=float32)
```

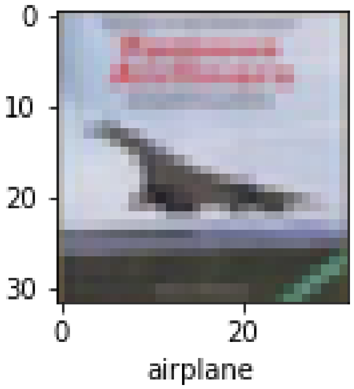
```
[55]: y_classes = [np.argmax(element) for element in y_pred]
      y_classes[:5]
```

```
[55]: [3, 1, 8, 0, 6]
```

```
[56]: y_test[:5]
```

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

```
[57]: plot_sample(x_test, y_test,3)
```



[58]:
```
classes[y_classes[3]]
```

[58]:
```
'airplane'
```

[ ]: