Image Classification Using CNN

September 5, 2023

1 Image Classification Using CNN

*CNN stands for **Convolutional Neural Network**. It's a type of deep learning algorithm that uses mathematical operations to analyze and process images. CNNs are often used for image classification and recognition because of their high accuracy.*

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

Load the dataset

[2]:
```
(X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
```

[3]:
```
X_test.shape
```

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

[4]:
```
X_train.shape
```

[4]:
```
(50000, 32, 32, 3)
```

[5]:
```
y_train.shape
```

[5]:
```
(50000, 1)
```

[6]:
```
y_train[:5]
```

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

[7]:
```
y_train = y_train.reshape(-1,)
y_train[:5]
```

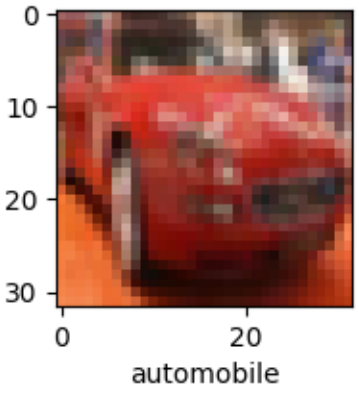
[7]:
```
array([6, 9, 9, 4, 1], dtype=uint8)
```

[8]:
```
y_test = y_test.reshape(-1,)
```

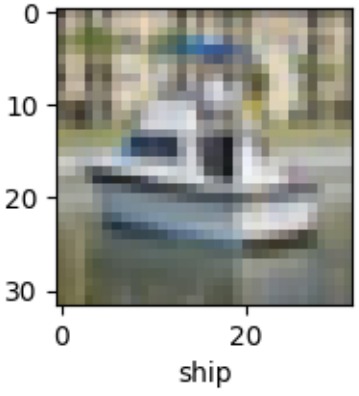
[9]:
```
classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',␣
↪'horse', 'ship', 'truck']
```

[10]:
```
def plot_sample(X, y, index):
    plt.figure(figsize=(15, 2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

[11]:
```
plot_sample(X_train, y_train, 5)
```



[12]:
```
plot_sample(X_train, y_train, 501)
```



Normalizing the training data

[13]:
```
X_train = X_train / 255.0
X_test = X_test / 255.0
```

Build simple artificial neural network for image classification

The code you have provided creates a sequential model with three dense layers. The first layer has 3000 neurons and uses the relu activation function. The second layer also has 3000 neurons and uses the relu activation function. The third layer has 10 neurons and uses the softmax activation function. The model is compiled using the SGD optimizer, the sparse_categorical_crossentropy loss function, and the accuracy metric. The model is then fit on the X_train and y_train data for 5 epochs.

[14]:
```
ann = models.Sequential([
    layers.Flatten(input_shape = (32, 32, 3)),
    layers.Dense(3000, activation = 'relu'),
    layers.Dense(3000, 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 [==============================] - 133s 84ms/step - loss: 1.8075 -
accuracy: 0.3568
Epoch 2/5
1563/1563 [==============================] - 150s 96ms/step - loss: 1.6185 -
accuracy: 0.4278
Epoch 3/5
1563/1563 [==============================] - 157s 101ms/step - loss: 1.5385 -
accuracy: 0.4579
Epoch 4/5
1563/1563 [==============================] - 157s 101ms/step - loss: 1.4790 -
accuracy: 0.4816
Epoch 5/5
1563/1563 [==============================] - 159s 102ms/step - loss: 1.4272 -
accuracy: 0.4987
```

[14]:
```
<keras.src.callbacks.History at 0x149134171f0>
```

[15]:
```
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))
```

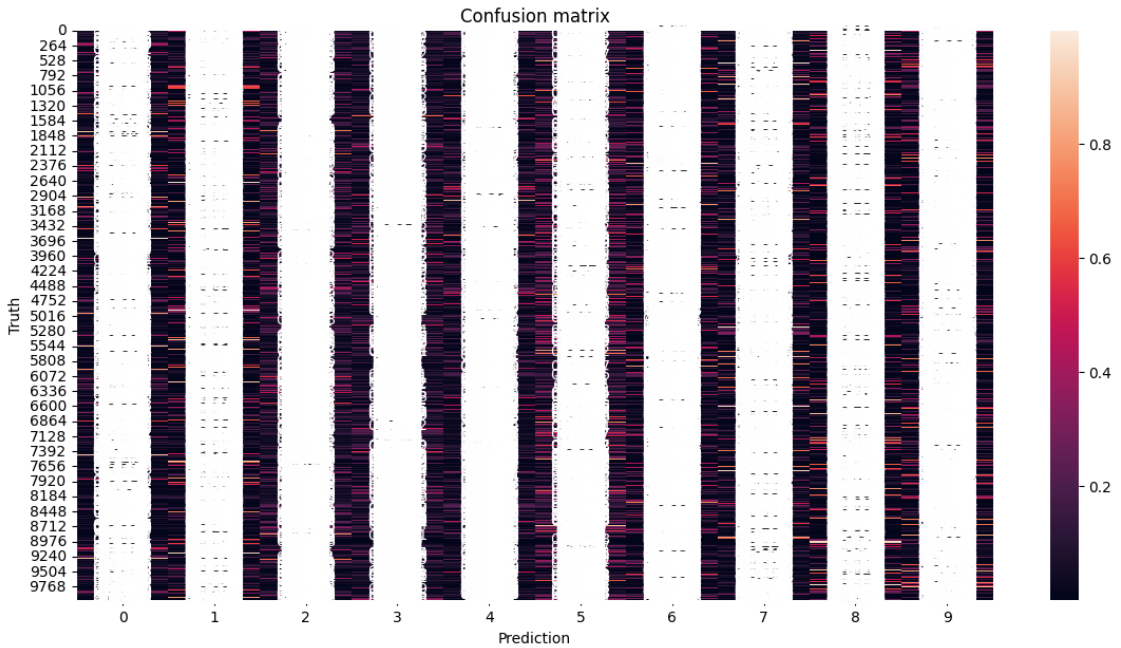
```
313/313 [==============================] - 8s 25ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.47      0.55      1000
           1       0.63      0.61      0.62      1000
           2       0.39      0.38      0.39      1000
           3       0.41      0.20      0.27      1000
           4       0.50      0.30      0.38      1000
           5       0.29      0.57      0.38      1000
           6       0.50      0.61      0.55      1000
           7       0.60      0.51      0.55      1000
           8       0.59      0.66      0.62      1000
           9       0.55      0.58      0.57      1000

    accuracy                           0.49     10000
   macro avg       0.51      0.49      0.49     10000
weighted avg       0.51      0.49      0.49     10000
```

[16]:
```
import seaborn as sns
```

[17]:
```
plt.figure(figsize = (14, 7))
sns.heatmap(y_pred, annot = True)
plt.ylabel('Truth')
plt.xlabel('Prediction')
plt.title('Confusion matrix')
plt.show()
```



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

[18]:
```
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')
])
```

[19]:
```
cnn.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[20]:
```
cnn.fit(X_train, y_train, epochs = 10)
```

```
Epoch 1/10
1563/1563 [==============================] - 27s 17ms/step - loss: 1.4759 -
accuracy: 0.4716
Epoch 2/10
1563/1563 [==============================] - 26s 17ms/step - loss: 1.1178 -
accuracy: 0.6091
Epoch 3/10
1563/1563 [==============================] - 26s 17ms/step - loss: 0.9900 -
accuracy: 0.6572
Epoch 4/10
1563/1563 [==============================] - 26s 17ms/step - loss: 0.8986 -
accuracy: 0.6882
Epoch 5/10
1563/1563 [==============================] - 26s 17ms/step - loss: 0.8317 -
accuracy: 0.7116
Epoch 6/10
1563/1563 [==============================] - 26s 17ms/step - loss: 0.7670 -
accuracy: 0.7313
Epoch 7/10
1563/1563 [==============================] - 27s 17ms/step - loss: 0.7192 -
accuracy: 0.7506
Epoch 8/10
1563/1563 [==============================] - 27s 17ms/step - loss: 0.6743 -
accuracy: 0.7635
Epoch 9/10
```

```
1563/1563 [==============================] - 31s 20ms/step - loss: 0.6283 -
accuracy: 0.7788
Epoch 10/10
1563/1563 [==============================] - 25s 16ms/step - loss: 0.5901 -
accuracy: 0.7933
```

[20]:
```
<keras.src.callbacks.History at 0x149ebf80550>
```

[21]:
```
cnn.evaluate(X_test, y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 1.0035 -
accuracy: 0.6881
```

[21]:
```
[1.0034927129745483, 0.6880999803543091]
```

[22]:
```
y_prediction = cnn.predict(X_test)
y_prediction[:5]
```

```
313/313 [==============================] - 2s 5ms/step
```

[22]:
```
array([[1.50252799e-05, 1.75855484e-06, 1.11156332e-04, 9.71733391e-01,
        1.19842833e-03, 2.03023721e-02, 1.45012527e-04, 5.79165971e-05,
        6.43389812e-03, 1.01380465e-06],
       [1.45523492e-02, 5.63232973e-02, 3.87647424e-06, 4.02662039e-07,
        4.57641647e-07, 5.48552759e-09, 5.46153949e-08, 1.12221687e-09,
        9.29095685e-01, 2.37675340e-05],
       [1.72431558e-01, 9.97117087e-02, 4.65344032e-03, 1.15202321e-02,
        2.47314945e-03, 1.00852968e-03, 4.97458328e-04, 4.36103233e-04,
        6.73745394e-01, 3.35224643e-02],
       [8.94762874e-01, 1.92736639e-04, 2.77030794e-03, 1.80506744e-04,
        1.38107018e-04, 5.76132288e-06, 4.04942584e-05, 3.02574813e-06,
        1.01885654e-01, 2.04512562e-05],
       [9.57179580e-08, 3.28610645e-06, 1.86614308e-03, 3.69130559e-02,
        7.72183165e-02, 9.11501236e-03, 8.61036360e-01, 2.37735985e-05,
        1.38198677e-02, 4.01568377e-06]], dtype=float32)
```

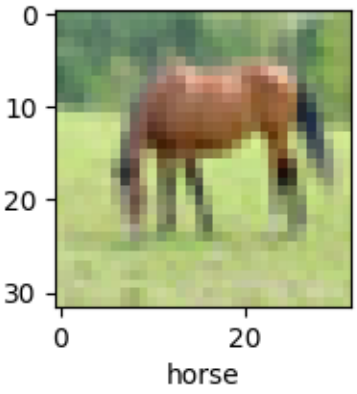
[23]:
```
y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

[23]:
```
[3, 9, 8, 8, 4]
```

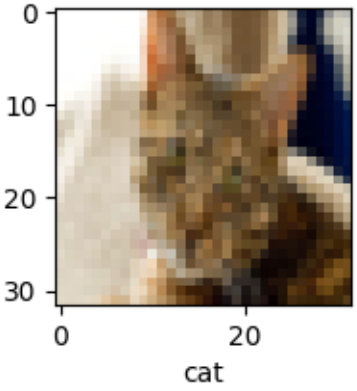
[24]:
```
y_test[:5]
```

[24]:
```
array([3, 8, 8, 0, 6], dtype=uint8)
```

[25]:
```
plot_sample(X_test, y_test, 60)
```



[26]:
```
plot_sample(X_test, y_test, 103)
```



[27]:
```
classes[y_classes[103]]
```

[27]: 'cat'

Written By-

Swapnil Bera