Experiment 06 : Write a program for Image Classification using CNN.

Learning Objective : Write a program for image classification using CNN.

Tools : Python

Theory :

CNN stands for Convolutional Neural Network. It's a type of deep learning algorithm that is particularly well-suited for analyzing visual data like images and videos. CNNs are inspired by the organization of the animal visual cortex and have been successful in various computer vision tasks such as image classification, object detection, segmentation, and more.

Here's how CNNs work :

1. **Convolutional Layers:** The core building blocks of CNNs are convolutional layers. These layers apply a set of learnable filters (also called kernels) to small patches of input data. This operation helps in detecting features like edges, textures, or patterns in the input images.
2. **Pooling Layers:** After convolutional layers, pooling layers are often added to reduce the spatial dimensions of the input volume, thus decreasing the computational complexity. Pooling layers typically perform operations like max-pooling or average-pooling to downsample the feature maps obtained from the convolutional layers.
3. **Activation Function:** Usually, after each convolutional or pooling layer, an activation function like ReLU (Rectified Linear Unit) is applied elementwise to introduce non-linearity into the network.
4. **Fully Connected Layers:** Towards the end of the network, one or more fully connected layers are added. These layers take the features extracted by the convolutional layers and learn to classify them into different categories based on their learned representations.
5. **Softmax Layer:** In classification tasks, a softmax layer is often used as the final layer to produce probability scores for each class. The class with the highest probability is chosen as the predicted class.

CNNs are typically trained using labeled datasets through a process called backpropagation, where the network adjusts its parameters (weights and biases) to minimize the difference between predicted and actual outputs.

CNNs have been instrumental in achieving state-of-the-art performance in various computer vision tasks and have found applications in diverse fields such as autonomous vehicles, medical image analysis, facial recognition, and more.

The basic steps to build an image classification model using a neural network are:

1. Flatten the input image dimensions to 1D (width pixels x height pixels).
2. Normalize the image pixel values (divide by 255).
3. One-Hot Encode the categorical column.
4. Build a model architecture (Sequential) with Dense layers (Fully connected layers).

Train the model and make predictions.

Implementation :

[1]:
```
import tensorflow as tf
from keras import datasets, layers, models
```

[2]:
```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.
↪load_data()

# Normalize pixel values between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

[3]:
```
# Print the shapes of the loaded data
print("Shape of training images:", train_images.shape)
print("Shape of training labels:", train_labels.shape)
print("Shape of test images:", test_images.shape)
print("Shape of test labels:", test_labels.shape)
```

```
Shape of training images: (50000, 32, 32, 3)
Shape of training labels: (50000, 1)
Shape of test images: (10000, 32, 32, 3)
Shape of test labels: (10000, 1)
```

[4]:
```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# Flatten the output from 2D to 1D
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))  # 10 classes for CIFAR-10

# Print a summary of the model architecture
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #

=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2  (None, 15, 15, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928
```

```
 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122570 (478.79 KB)
Trainable params: 122570 (478.79 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[5]:
```
model.compile(optimizer='adam', loss=tf.keras.losses.
  ↪SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=50,␣
  ↪validation_data=(test_images, test_labels))
```

```
Epoch 1/50
1563/1563 [==============================] - 17s 8ms/step - loss: 1.5257 -
accuracy: 0.4438 - val_loss: 1.2574 - val_accuracy: 0.5542
Epoch 2/50
1563/1563 [==============================] - 9s 6ms/step - loss: 1.1554 -
accuracy: 0.5900 - val_loss: 1.0552 - val_accuracy: 0.6211
Epoch 3/50
1563/1563 [==============================] - 7s 4ms/step - loss: 1.0114 -
accuracy: 0.6447 - val_loss: 0.9928 - val_accuracy: 0.6518
Epoch 48/50
1563/1563 [==============================] - 8s 5ms/step - loss: 0.1327 -
accuracy: 0.9532 - val_loss: 2.4169 - val_accuracy: 0.6766
Epoch 49/50
1563/1563 [==============================] - 8s 5ms/step - loss: 0.1232 -
accuracy: 0.9566 - val_loss: 2.4369 - val_accuracy: 0.6872
Epoch 50/50
1563/1563 [==============================] - 8s 5ms/step - loss: 0.1292 -
accuracy: 0.9570 - val_loss: 2.5471 - val_accuracy: 0.6754
```

[6]:
```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print('Test accuracy:', test_acc)
```

```
313/313 - 1s - loss: 2.5471 - accuracy: 0.6754 - 713ms/epoch - 2ms/step
Test accuracy: 0.6754000186920166
```

[7]:
```
model.save('model.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

[8]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from google.colab.patches import cv2_imshow
```

[9]:
```
# Load the trained model
model = tf.keras.models.load_model('model.h5')
```

[18]:
```
# Load and preprocess a test image
image_path = '/content/drive/MyDrive/Dataset/ImageSet/dog1.jpg'
image = tf.keras.preprocessing.image.load_img(image_path, target_size=(32, 32))
input_array = tf.keras.preprocessing.image.img_to_array(image)
input_array = np.expand_dims(input_array, axis=0)
input_array = input_array / 255.0
```

[19]:
```
img = cv2.imread(image_path)
cv2_imshow(img)
```

[20]:
```
# Make predictions
predictions = model.predict(input_array)
predicted_class = np.argmax(predictions[0])
```

```
1/1 [==============================] - 0s 30ms/step
```

[21]:
```
# Print the predicted class and confidence value
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',␣
↪'horse', 'ship', 'truck']
print("Predicted class:", class_names[predicted_class])
```

```
Predicted class: dog
```

Result and Discussion :

Learning Outcomes : Students should have the ability to

LO 6.1: Ability to implement image classification tasks using CNNs, where the network learns to classify images into different categories based on the features it extracts.
LO 6.2: Understand how CNNs can be adapted for object detection tasks, where the goal is to locate and classify objects within images.

Course Outcomes :

CO : Understand and apply Convolutional Neural Networks.

Conclusion :

_____

_____

Viva Questions :

Q1. What is the purpose of activation functions in CNNs, and which activation functions are commonly used?
Q2. How do convolutional layers work in a CNN, and what role do they play in image classification?

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	Total
Marks Obtained				