Ex: 3 b

BUILD A CONVOLUTIONAL NEURAL NETWORK

Aim:

To build a simple convolutional neural network without Keras/TensorFlow.

Algorithm:

1. Define Activation Functions: Create ReLU and softmax activation functions for non-linear transformations.
2. Implement Convolution: Define a 2D convolution function that slides a kernel over the input image and computes the dot product at each position.
3. Apply Convolution Layer: Initialize a convolution kernel, apply the convolution function, and use ReLU activation on the result.
4. Implement Pooling: Define a max-pooling function to reduce the spatial dimensions by selecting the maximum value in each sub-region.
5. Flatten and Fully Connect: Flatten the pooled output and apply a fully connected layer with weights, calculating logits.
6. Output Using Softmax: Apply the softmax function to logits for final class probabilities.

Program:

```
import numpy as np
def relu(x):
    return np.maximum(0, x)
def softmax(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / np.sum(exp_x, axis=0)

# Convolution operation
def convolve2d(X, kernel):
    kernel_height, kernel_width = kernel.shape
    height, width = X.shape
```

```
    output = np.zeros((height - kernel_height + 1, width - kernel_width + 1))
    for i in range(output.shape[0]):
        for j in range(output.shape[1]):
            output[i, j] = np.sum(X[i:i+kernel_height, j:j+kernel_width] * kernel)
    return output

# Pooling operation
def max_pooling(X, size=2, stride=2):
    output_height = (X.shape[0] - size) // stride + 1
    output_width = (X.shape[1] - size) // stride + 1
    output = np.zeros((output_height, output_width))
    for i in range(output_height):
        for j in range(output_width):
            output[i, j] = np.max(X[i*stride:i*stride+size, j*stride:j*stride+size])
    return output

def simple_cnn(X):
    # Convolution Layer
    kernel = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])  # Example kernel
    conv_output = convolve2d(X, kernel)
    conv_output = relu(conv_output)
    # Pooling Layer
    pooled_output = max_pooling(conv_output)
    # Flatten and Fully Connected Layer
    flattened = pooled_output.flatten()
    weights = np.random.randn(flattened.size, 10)  # Assuming 10 classes
    logits = np.dot(flattened, weights)
    output = softmax(logits)
    return output
# Example input data (a 28x28 grayscale image)
```

```
X = np.random.rand(28, 28)
output = simple_cnn(X)
print("Output:", output)
```

Output:

```
Output: [1.76093045e-07 2.85912698e-14 1.66949370e-16 1.76633408e-07
 1.19544489e-13 4.08544451e-06 5.86145334e-06 2.86661537e-05
 4.93999040e-01 5.05961994e-01]
```

Result:

Thus the program to build a simple convolutional neural network without Keras/TensorFlow has been executed successfully.