Assignment on AlexNet architecture.

MTech in Applied AI
Natural Language Processing

by

Sanjeev Kumar Pandey[*]
Faculty: Dr Saugata Sinha

Submitted April 20, 2025

[*] Student ID: 31050, Enrolment No:MT23AAI001, Email: pandey.sanjeev@yahoo.com

Table of Contents

Assignment 2

1 Instructions

Part 1

Write a function for performing the operations in the convolution layer, in a convolution layer of a CNN. As it is mentioned in the class, the function should be written in a general manner i.e., it should be able to handle the actual convolution/cross-correlation with kernels of any size, any stride. It should also accommodate an activation function (ReLU) after cross-correlation/convolution.

Part 2

Write another function for performing the operations in the pooling layer, of a CNN. As it is mentioned in the class, the function should be written in a general manner i.e., it should be able to handle the actual pooling (max or average, you should make provisions for both) with kernels of any size, any stride.

Part 3

Write another function for performing the operations in the Dense layer, of a CNN. As it is mentioned in the class, the function should be written in a general manner i.e., it should be able to handle forward propagation for any number of nodes and a bias. It should also accommodate an activation function (ReLU) after cross-correlation/convolution.

Part 4

Use that function to build the AlexNet architecture. (You only need to build the architecture, you do not need to train it.)

In your report, please describe the detailed architecture of AlexNet that you have considered. You only need to submit the Python code.

Chapter 1

Part 1 - Convolution operation in a CNN

1 Part 1 - Solution and explanation

1.1 Python code - Part 1 - Convolution operation in a CNN

```python
import numpy as np

class ConvLayer:
    def __init__(self, num_filters, kernel_size=3, stride=1, padding=0):
        self.num_filters = num_filters
        self.kernel_size = kernel_size
        self.stride = stride
        self.padding = padding
        self.kernels = np.random.randn(num_filters, kernel_size, kernel_size) * 0.1


    def relu(self, x):
        return np.maximum(0, x)

    def forward(self, x):
        print(f'ConvLayer : Input Matrix:\n{x}')
        if x.ndim == 2:
            x = np.expand_dims(x, axis=-1)

        input_h, input_w, input_c = x.shape

        if self.padding > 0:
            x_padded = np.pad(x,
                              ((self.padding, self.padding),
                               (self.padding, self.padding),
                               (0, 0)),
                              mode='constant')
        else:
            x_padded = x

        output_h = (input_h + 2*self.padding - self.kernel_size) // self.stride + 1
        output_w = (input_w + 2*self.padding - self.kernel_size) // self.stride + 1

        output = np.zeros((output_h, output_w, self.num_filters))

        for f in range(self.num_filters):
            kernel = self.kernels[f]
```

```
            for y in range(output_h):
                for x_pos in range(output_w):
                    y_start = y * self.stride
                    y_end = y_start + self.kernel_size
                    x_start = x_pos * self.stride
                    x_end = x_start + self.kernel_size

                    region = x_padded[y_start:y_end, x_start:x_end, :]
                 #   print(f'region : {region}')
                    output[y, x_pos, f] = np.sum(region * kernel[:, :, np.newaxis])
                 #   print(f'output[{y}, {x_pos}, {f}] : {output[y, x_pos, f]}')

        print(f'ConvLayer before ReLU output : {output}')
        return self.relu(output)
```

Listing 1.1: Code

1.2 Python code - Part 1 - Convolution operation in a CNN - Usage

```
# Testing ConvLayer with small 4x4 input
import numpy as np
from MT23AAI001_assignment_2_part_1 import ConvLayer

# Create a simple input
input_matrix = np.array([
    [1, 2, 0, 1],
    [3, 1, 2, 2],
    [0, 1, 3, 1],
    [2, 2, 1, 0]
])

# Create ConvLayer (1 filter, 2x2 kernel, stride 1, padding 0)
conv_layer = ConvLayer(num_filters=1, kernel_size=2, stride=1, padding=0)

# Forward pass
output = conv_layer.forward(input_matrix)

print("Convolution Output:")
print(output)  # remove last dimension for clarity
print(f"final ConvLayer output {output[:, :, 0]}")  # remove last dimension for clarity
```

Listing 1.2: Code

1.3 Output - Part 1 - Convolution operation in a CNN

```
C:\Users\urssa\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\sub
ConvLayer : Input Matrix:
[[1 2 0 1]
 [3 1 2 2]
 [0 1 3 1]
 [2 2 1 0]]
ConvLayer before ReLU output : [[[-0.38151255]
  [-0.30066298]
  [-0.35989712]]

 [[ 0.02868029]
  [-0.14325093]
  [-0.38775583]]

 [[-0.35989712]
  [-0.10227455]
  [-0.09045052]]]
Convolution Output:
[[[0.        ]
  [0.        ]
```

```
 [0.        ]]

[[0.02868029]
 [0.        ]
 [0.        ]]

[[0.        ]
 [0.        ]
 [0.        ]]]
final ConvLayer output [[0.          0.          0.        ]
 [0.02868029 0.          0.        ]
 [0.         0.          0.        ]]

Process finished with exit code 0
```

Listing 1.3: Convolution operation in a CNN

Explanation of code and operation

1. Input and Configuration

Given a 4×4 input matrix:

$$X = \begin{bmatrix} 1 & 2 & 0 & 1 \\ 3 & 1 & 2 & 2 \\ 0 & 1 & 3 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

And a convolution layer with the following configuration:

- Number of filters: 1
- Kernel size: 2×2
- Stride: 1
- Padding: 0

2. Output Dimensions

Given:

$$\text{Input size} = H \times W = 4 \times 4, \quad \text{Kernel size} = K = 2, \quad \text{Stride} = S = 1, \quad \text{Padding} = P = 0$$

The output height and width are computed as:

$$\text{Output Height} = \left\lfloor \frac{H + 2P - K}{S} \right\rfloor + 1 = \left\lfloor \frac{4 + 0 - 2}{1} \right\rfloor + 1 = 3$$

$$\text{Output Width} = \left\lfloor \frac{W + 2P - K}{S} \right\rfloor + 1 = \left\lfloor \frac{4 + 0 - 2}{1} \right\rfloor + 1 = 3$$

So the output is a 3×3 feature map.

3. Convolution Process

Assume a sample kernel:

$$K = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}$$

We slide the kernel over the input matrix and compute dot products:

Example: Top-left Region

Take the top-left 2×2 patch of X:

$$\text{Region} = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$$

Compute the element-wise product and sum:

$$(1)(0.1) + (2)(0.2) + (3)(0.3) + (1)(0.4) = 0.1 + 0.4 + 0.9 + 0.4 = 1.8$$

This value becomes the top-left element of the output.

Repeat for All Positions

Repeat the above process for each valid 2×2 patch in the input. This results in a 3×3 output matrix.

4. ReLU Activation

The output of convolution is passed through the ReLU activation function:

$$\text{ReLU}(x) = \max(0, x)$$

Any negative values are replaced by 0.

5. Final Output

Let the final convolution output (before ReLU) be:

$$Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{bmatrix} \quad \Rightarrow \quad \text{ReLU}(Z) = \begin{bmatrix} \max(0, z_{11}) & \max(0, z_{12}) & \max(0, z_{13}) \\ \max(0, z_{21}) & \max(0, z_{22}) & \max(0, z_{23}) \\ \max(0, z_{31}) & \max(0, z_{32}) & \max(0, z_{33}) \end{bmatrix}$$

Why Padding?

If no padding was used, after convolution the output would shrink. Padding is added to preserve spatial size or to control the output size.

Final important point:

- The "regions" selected are the current 2×2 window from the input matrix.
- The kernel applies element-wise multiplication and addition over that window.
- The resulting sum is stored in the output matrix at the corresponding location (i, j).
- After that, ReLU activation is applied: negative outputs are replaced with 0.

Full Visual Idea:

At each move:

1. Slide the kernel across the input (step size = stride).
2. Perform element-wise multiplication and sum the results.
3. Store the result into the output matrix.
4. Apply ReLU to the stored value.

Chapter 2

Part 2 - Operations in the pooling layer, of a CNN

1 Part 2 - Solution and explanation

1.1 Python code - Part 2 - Operations in the pooling layer, of a CNN

```
import numpy as np

class PoolingLayer:
    def __init__(self, kernel_size=2, stride=2, pooling_type='max', padding=0):
        self.kernel_size = (kernel_size, kernel_size)
        self.stride = (stride, stride)
        self.pooling_type = pooling_type
        self.padding = padding

    def forward(self, input_matrix):
        if self.padding > 0:
            input_padded = np.pad(input_matrix,
                                  ((self.padding, self.padding),
                                   (self.padding, self.padding),
                                   (0, 0)),
                                  mode='constant')
        else:
            input_padded = input_matrix

        kernel_h, kernel_w = self.kernel_size
        stride_h, stride_w = self.stride
        input_h, input_w, input_c = input_padded.shape

        out_h = (input_h - kernel_h) // stride_h + 1
        out_w = (input_w - kernel_w) // stride_w + 1

        output = np.zeros((out_h, out_w, input_c))

        for c in range(input_c):
            for y in range(out_h):
                for x in range(out_w):
                    y_start = y * stride_h
                    y_end = y_start + kernel_h
                    x_start = x * stride_w
                    x_end = x_start + kernel_w

                    window = input_padded[y_start:y_end, x_start:x_end, c]
```

```

                    if self.pooling_type == 'max':
                        output[y, x, c] = np.max(window)
                    elif self.pooling_type == 'average':
                        output[y, x, c] = np.mean(window)
                    else:
                        raise ValueError("Pooling type must be 'max' or 'average'")

        return output
```

Listing 2.1: Code

1.2 Python code - Part 2 - Operations in the pooling layer, of a CNN - Usage

```
# Testing PoolingLayer with small 4x4x1 input
import numpy as np
from MT23AAI001_assignment_2_part_2 import PoolingLayer
from MT23AAI001_assignment_2_part_1_usage import output
# Create a simple input (adding channel dimension)
input_matrix = output

# Create PoolingLayer (2x2 kernel, stride 2, max pooling)
pool_layer = PoolingLayer(kernel_size=2, stride=2, pooling_type='max')

# Forward pass
output = pool_layer.forward(input_matrix)

print("Pooling Output:")
print(output[:, :, 0])  # remove last dimension for clarity
```

Listing 2.2: Code

1.3 Output - Part 2 - Operations in the pooling layer, of a CNN

```
C:\Users\urssa\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\sub
ConvLayer : Input Matrix:
[[1 2 0 1]
 [3 1 2 2]
 [0 1 3 1]
 [2 2 1 0]]
ConvLayer before ReLU output : [[[0.35787586]
  [0.36055814]
  [0.08085561]]

 [[0.56384506]
  [0.24854975]
  [0.53751799]]

 [[0.08085561]
  [0.36372426]
  [0.61850811]]]
Convolution Output:
[[[0.35787586]
  [0.36055814]
  [0.08085561]]

 [[0.56384506]
  [0.24854975]
  [0.53751799]]

 [[0.08085561]
  [0.36372426]
  [0.61850811]]]
final ConvLayer output [[0.35787586 0.36055814 0.08085561]
 [0.56384506 0.24854975 0.53751799]
```

```
 [0.08085561 0.36372426 0.61850811]]
Pooling Output:
[[0.56384506]]

Process finished with exit code 0
```

Listing 2.3: Convolution operation in a CNN

Chapter 3

Part 3 - Operations in the Dense layer, of a CNN

1 Part 3 - Solution and explanation

1.1 Python code - Part 3 - Operations in the Dense layer, of a CNN

```
import numpy as np
from MT23AAI001_assignment_2_part_1 import ConvLayer
from MT23AAI001_assignment_2_part_2 import PoolingLayer

class DenseLayer:
    def __init__(self, input_size, output_size):
        self.weights = np.random.randn(input_size, output_size) * np.sqrt(2. / input_size)
        self.bias = np.zeros((1, output_size))

    def relu(self, x):
        return np.maximum(0, x)

    def forward(self, x):
        original_shape = x.shape
        if x.ndim > 1:
            x = x.reshape(original_shape[0], -1)
        else:
            x = x.reshape(1, -1)

        linear_output = np.dot(x, self.weights) + self.bias
        activated = self.relu(linear_output)
        return activated.squeeze()

if __name__ == "__main__":
    input_image = np.random.randn(32, 32, 3)

    conv_layer = ConvLayer(num_filters=8, kernel_size=3, stride=1, padding=1)
    pool_layer = PoolingLayer(kernel_size=2, stride=2, pooling_type='max')

    x = conv_layer.forward(input_image)
    x = pool_layer.forward(x)
    x = x.flatten()

    dense_layer = DenseLayer(input_size=x.size, output_size=10)

    output = dense_layer.forward(x)
```

```
    print("Output vector:", output)
```

Listing 3.1: Code

1.2 Python code - Part 3 - Operations in the Dense layer, of a CNN - Usage

```
# Testing DenseLayer with small input
import numpy as np
from MT23AAI001_assignment_2_part_3 import DenseLayer
from MT23AAI001_assignment_2_part_1 import ConvLayer
from MT23AAI001_assignment_2_part_2 import PoolingLayer

# Create a small flattened input (say 6 features)
input_image = np.array([
    [1, 2, 0, 1],
    [3, 1, 2, 2],
    [0, 1, 3, 1],
    [2, 2, 1, 0]
])

conv_layer = ConvLayer(num_filters=8, kernel_size=3, stride=1, padding=1)
pool_layer = PoolingLayer(kernel_size=2, stride=2, pooling_type='max')

x = conv_layer.forward(input_image)
print(f'conv output : {x}')
x = pool_layer.forward(x)
print(f'maxpool output : {x}')

x = x.flatten()

dense_layer = DenseLayer(input_size=x.size, output_size=10)

output = dense_layer.forward(x)
print(f"Dense layer Output vector:{output}")
```

Listing 3.2: Code

1.3 Output - Part 3 - Operations in the Dense layer, of a CNN

```
C:\Users\urssa\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\sub
ConvLayer : Input Matrix:
[[1 2 0 1]
 [3 1 2 2]
 [0 1 3 1]
 [2 2 1 0]]
ConvLayer before ReLU output : [[[-0.00368663 -0.34108194  0.63824441 -0.23205162 -0.01325644
    0.45929073  0.20682844 -0.37103491]
  [-0.65818436  0.48162039  0.05321291  0.15868699  0.14877946
   -0.24458851 -0.1445004   0.13010769]
  [ 0.00995703 -0.31878218  0.32486327 -0.15137688  0.2175723
    0.17159034  0.31279352 -0.46176828]
  [-0.39845898  0.20694496  0.13772996  0.25492959  0.19477609
   -0.13412364  0.07934083  0.1443939 ]]

 [[-0.99410932  0.51064619  0.09265284 -0.27086462 -0.19990808
   -0.06561158 -0.18557474  0.58788721]
  [ 0.02658353 -0.27458023  0.36534869 -0.66009987 -0.37791086
    0.43077029  0.22733217 -0.59080128]
  [-0.7811821  -0.11303787  0.55202826  0.12600403 -0.06315362
   -0.02109283  0.3487565   0.19668696]
  [-0.65656998  0.61822023 -0.09106233  0.14871651  0.12332346
   -0.41410606 -0.02613262  0.27473329]]

 [[ 0.05873378 -0.41156634  0.19199836 -0.79519402 -0.31634094
    0.62007236  0.25708232 -0.25589561]
```

```
  [-0.71664707 -0.53260648  0.1607009  -0.4804024  -0.38993081
   -0.11669148  0.19344265  0.66995408]
  [-1.21753019  0.56651465 -0.06386178 -0.15694042 -0.19632238
   -0.37386843 -0.00218271  0.85113841]
  [-0.44009038  0.30518895 -0.10775859  0.10294759 -0.277023
   -0.30163473  0.25568678  0.12903751]]

 [[-0.44343381  0.13535475  0.15395094 -0.3697194  -0.17878913
   -0.02814721 -0.22257599  0.34781909]
  [-1.10280907  0.34559906 -0.095137   -0.2506273   0.11148951
   -0.38574896  0.15781001  0.74832115]
  [-0.45242563  0.2699511  -0.14668508 -0.26198116 -0.33632749
   -0.05893645  0.23018783  0.22690705]
  [-0.25567124 -0.08390465 -0.0076355   0.24545514 -0.3933043
   -0.16373322  0.28128908  0.17278697]]]
conv output : [[[0.         0.         0.63824441 0.         0.         0.45929073
   0.20682844 0.        ]
  [0.         0.48162039 0.05321291 0.15868699 0.14877946 0.
   0.         0.13010769]
  [0.00995703 0.         0.32486327 0.         0.2175723  0.17159034
   0.31279352 0.        ]
  [0.         0.20694496 0.13772996 0.25492959 0.19477609 0.
   0.07934083 0.1443939 ]]

 [[0.         0.51064619 0.09265284 0.         0.         0.
   0.         0.58788721]
  [0.02658353 0.         0.36534869 0.         0.         0.43077029
   0.22733217 0.        ]
  [0.         0.         0.55202826 0.12600403 0.         0.
   0.3487565  0.19668696]
  [0.         0.61822023 0.         0.14871651 0.12332346 0.
   0.         0.27473329]]

 [[0.05873378 0.         0.19199836 0.         0.         0.62007236
   0.25708232 0.        ]
  [0.         0.         0.1607009  0.         0.         0.
   0.19344265 0.66995408]
  [0.         0.56651465 0.         0.         0.         0.
   0.         0.85113841]
  [0.         0.30518895 0.         0.10294759 0.         0.
   0.25568678 0.12903751]]

 [[0.         0.13535475 0.15395094 0.         0.         0.
   0.         0.34781909]
  [0.         0.34559906 0.         0.         0.11148951 0.
   0.15781001 0.74832115]
  [0.         0.2699511  0.         0.         0.         0.
   0.23018783 0.22690705]
  [0.         0.         0.         0.24545514 0.         0.
   0.28128908 0.17278697]]]
maxpool output : [[[0.02658353 0.51064619 0.63824441 0.15868699 0.14877946 0.45929073
   0.22733217 0.58788721]
  [0.00995703 0.61822023 0.55202826 0.25492959 0.2175723  0.17159034
   0.3487565  0.27473329]]

 [[0.05873378 0.34559906 0.19199836 0.         0.11148951 0.62007236
   0.25708232 0.74832115]
  [0.         0.56651465 0.         0.24545514 0.         0.
   0.28128908 0.85113841]]]
Dense layer Output vector:[0.33771126 0.06349989 0.         0.         0.07122175 0.20331102
 0.         0.         0.         0.        ]

Process finished with exit code 0
```

Listing 3.3: Operations in the Dense layer of a CNN

Chapter 4

Part 4 - AlexNet architecture

AlexNet Architecture (Using ConvLayer, PoolingLayer, DenseLayer)

Detailed Architecture Description

- **Input:** $227 \times 227 \times 3$
- **Conv1:** 96 filters, 11×11 kernel, stride 4, padding 0 $\rightarrow 55 \times 55 \times 96$
- **Pool1:** Max Pooling, 3×3 kernel, stride 2 $\rightarrow 27 \times 27 \times 96$
- **Conv2:** 256 filters, 5×5 kernel, stride 1, padding 2 $\rightarrow 27 \times 27 \times 256$
- **Pool2:** Max Pooling, 3×3 kernel, stride 2 $\rightarrow 13 \times 13 \times 256$
- **Conv3:** 384 filters, 3×3 kernel, stride 1, padding 1 $\rightarrow 13 \times 13 \times 384$
- **Conv4:** 384 filters, 3×3 kernel, stride 1, padding 1 $\rightarrow 13 \times 13 \times 384$
- **Conv5:** 256 filters, 3×3 kernel, stride 1, padding 1 $\rightarrow 13 \times 13 \times 256$
- **Pool3:** Max Pooling, 3×3 kernel, stride 2 $\rightarrow 6 \times 6 \times 256$
- **Flatten:** $6 \times 6 \times 256 = 9216$ features
- **Dense1:** 4096 neurons
- **Dense2:** 4096 neurons
- **Dense3:** 10 neurons (number of classes)

1 Part 3 - Solution and explanation

1.1 Python code - Part 4 - AlexNet architecture

```
import numpy as np
from MT23AAI001_assignment_2_part_1 import ConvLayer
from MT23AAI001_assignment_2_part_2 import PoolingLayer
from MT23AAI001_assignment_2_part_3 import DenseLayer

class AlexNet:
    def __init__(self, num_classes=10):
```

```
        """
        Build the AlexNet architecture
        """
        self.conv1 = ConvLayer(num_filters=96, kernel_size=11, stride=4, padding=0)
        self.pool1 = PoolingLayer(kernel_size=3, stride=2, pooling_type='max')

        self.conv2 = ConvLayer(num_filters=256, kernel_size=5, stride=1, padding=2)
        self.pool2 = PoolingLayer(kernel_size=3, stride=2, pooling_type='max')

        self.conv3 = ConvLayer(num_filters=384, kernel_size=3, stride=1, padding=1)
        self.conv4 = ConvLayer(num_filters=384, kernel_size=3, stride=1, padding=1)
        self.conv5 = ConvLayer(num_filters=256, kernel_size=3, stride=1, padding=1)
        self.pool3 = PoolingLayer(kernel_size=3, stride=2, pooling_type='max')

        # Dense layers
        self.fc1 = DenseLayer(input_size=6*6*256, output_size=4096)
        self.fc2 = DenseLayer(input_size=4096, output_size=4096)
        self.fc3 = DenseLayer(input_size=4096, output_size=num_classes)

    def forward(self, x):
        """
        Forward pass through AlexNet
        """
        x = self.conv1.forward(x)
        x = self.pool1.forward(x)

        x = self.conv2.forward(x)
        x = self.pool2.forward(x)

        x = self.conv3.forward(x)
        x = self.conv4.forward(x)
        x = self.conv5.forward(x)
        x = self.pool3.forward(x)

        x = x.flatten()

        x = self.fc1.forward(x)
        x = self.fc2.forward(x)
        x = self.fc3.forward(x)

        return x

if __name__ == "__main__":
    # Example input: (227, 227, 3) image as in original AlexNet
    input_image = np.random.randn(227, 227, 3)

    model = AlexNet(num_classes=10)
    output = model.forward(input_image)

    print("Output vector:", output)
```

Listing 4.1: Code

1.2 Output - Part 4 - AlexNet architecture

```
C:\Users\urssa\AppData\Local\Microsoft\WindowsApps\python3.11.exe C:\Sanjeev\VNIT_CLASSES\VNIT-AAI-SEM3\sub
ConvLayer : Input Matrix:
[[[ 2.65693186e-01 -6.21752486e-01  1.89641706e+00]
  [ 5.95163442e-01  6.38419762e-02  8.94719502e-01]
  [-1.13291501e+00 -3.03471912e-01 -4.70213452e-01]
  ...
  [-6.01857817e-01 -1.00141111e-01  2.54749786e-01]
  [ 4.43728329e-01  1.81751681e+00  2.92888807e-01]
  [-9.66107872e-01 -1.21946076e+00 -2.05127713e-01]]

```

```
[[ 5.26016853e-01 -6.55529495e-02  9.98617744e-01]
  [-3.37120923e-02 -5.42978666e-01  7.21488580e-01]
  [ 3.89172826e-02  2.05491413e-01  1.50705835e-01]
  ...
  [ 1.33530508e-03 -8.53893455e-02 -1.39191479e+00]
  [-8.38717743e-01 -6.10260849e-02  3.86556049e-02]
  [-1.82610880e-01 -1.56710798e+00 -8.71169007e-01]]

 [[-6.67399205e-01 -1.52279139e+00  1.10503491e+00]
  [ 7.47793975e-01  3.04773556e-01 -2.46866403e+00]
  [-1.42562100e+00 -1.44081288e+00 -2.28698741e-01]
  ...
  [ 1.01518636e+00 -8.04498033e-02  1.58861141e-01]
  [ 1.78784058e-02 -1.90620937e-01 -1.49872605e+00]
  [ 6.97840227e-01 -1.70499895e+00 -2.39549188e+00]]

 ...

 [[-6.36761928e-01  2.66592204e-01  1.20971910e+00]
  [ 5.42213420e-02 -1.66049207e+00 -1.02347575e+00]
  [-7.89599434e-01  1.91871092e+00  3.89949848e-02]
  ...
  [ 7.79281613e-01 -4.73125397e-01  8.07596784e-01]
  [-4.85654474e-02 -6.07978271e-01  1.33521932e+00]
  [ 1.34427354e+00  8.41086311e-01  6.26665081e-01]]

 [[ 1.77358185e+00  8.52722897e-01  6.92284511e-01]
  [-7.83291795e-02 -8.96914013e-01 -1.48291344e-01]
  [ 5.28139309e-01 -1.50624632e+00 -1.16776735e+00]
  ...
  [ 3.82051561e-01 -6.79442754e-01 -2.45136236e-01]
  [-1.59901366e+00  9.00276765e-01  2.03683322e+00]
  [ 4.15607160e-01 -7.61809427e-01  2.79883049e-01]]

 [[-4.54848328e-01 -1.43917529e+00  2.49348335e+00]
  [-1.51530771e+00 -4.47950485e-02  8.20474529e-02]
  [-3.13461970e-02  2.80542439e-01  5.00152155e-01]
  ...
  [ 1.98906800e+00 -1.68469905e+00  7.64384071e-01]
  [-9.86336414e-01 -3.75438237e-01  1.30309580e+00]
  [ 3.84514296e-02  4.45046905e-01 -1.01666172e+00]]]
ConvLayer before ReLU output : [[[ 1.34581297e+00 -1.33060798e+00  1.71490278e+00 ...  1.29513126e+00
   -2.01603684e+00 -3.32869659e+00]
  [ 1.71131229e+00  3.32021319e+00  2.55952439e+00 ... -9.57823634e-02
   -1.71550637e+00 -7.40630387e-01]
  [ 1.82285285e+00 -2.10217852e+00  1.99354477e+00 ... -1.79190694e+00
   -3.66286911e-01  1.52882223e+00]
  ...
  [ 1.95502874e-01  6.87208783e-01  2.11363837e+00 ...  6.80334771e-01
   -2.09439146e-01 -2.40759488e+00]
  [ 1.22065336e-01  1.77815851e+00  3.41787332e-02 ... -8.53408449e-01
   -3.03661086e+00  2.01509478e-01]
  [-8.36817866e-01  1.66975643e+00  3.26128984e+00 ...  2.90454729e-01
   -6.74837167e-01 -1.36769404e+00]]

 [[ 6.30758660e-01  1.17357429e-01  1.85574208e+00 ...  2.59298021e+00
    2.37246782e-01  3.02355589e+00]
  [ 6.06933958e-01 -4.89464040e-01 -6.93081386e-01 ... -1.68089880e+00
   -1.02415467e+00 -1.85134836e+00]
  [ 3.75743942e-01 -6.40452069e-01  2.06710698e+00 ...  1.37107900e+00
   -6.61037515e-01 -1.30103574e+00]
  ...
  [ 2.18226735e+00 -9.32602200e-01 -3.20130209e-01 ...  1.43645800e+00
    1.09817330e+00 -7.05121527e-01]
  [ 6.83052645e-01 -6.13881124e-01 -2.00776315e+00 ... -1.00935491e-02
```

```
   3.45377324e+00 -1.46695855e+00]
 [ 2.91878784e-02  2.48238967e+00  4.15478498e-01 ...  1.98636755e+00
  -1.58170673e+00 -7.74972267e-01]]

[[ 2.78290873e+00  2.68988657e+00  1.71162597e+00 ...  1.69153927e-01
  -1.71195485e-01 -9.57179318e-02]
 [-2.82918537e+00  1.67590547e+00 -3.64297275e+00 ...  8.83094764e-01
  -1.65114131e+00 -1.56439539e+00]
 [-2.73260022e-01 -2.05131526e+00  3.36315089e+00 ...  1.10200259e+00
   2.58794124e+00 -1.07642710e-01]
 ...
 [ 1.53218888e+00  5.65853154e-01  1.59872353e+00 ... -4.00907034e+00
  -2.02722106e+00  2.31183329e+00]
 [ 5.24127218e-01  1.39147118e+00 -7.68216147e-01 ...  1.21717909e+00
   4.18489315e-01  3.96791644e+00]
 [-2.37845606e+00  6.26854787e-01 -2.19124311e+00 ... -1.52534865e+00
  -2.10751324e+00 -2.87783393e-01]]

...

[[ 1.89100608e+00  2.80823687e+00  9.52467744e-01 ...  5.96909018e-01
  -1.99975217e+00  1.02447563e+00]
 [-1.34712178e+00  2.72509410e+00  9.35344404e-01 ... -4.36609036e+00
  -2.52211660e-01  1.82424301e+00]
 [-2.64561557e+00 -9.59096911e-01 -1.42425018e+00 ... -2.68669544e+00
  -6.73616090e-01  1.08490623e+00]
 ...
 [ 1.05646672e+00  1.64273670e-01 -9.42800364e-01 ...  2.50363526e+00
   3.86871359e-01  8.31704505e-01]
 [ 2.05732462e+00 -1.13821512e+00 -8.00863449e-02 ... -3.05397801e+00
  -2.62233884e+00  2.30858144e+00]
 [-7.54865032e-02 -1.58085042e+00 -2.99678304e+00 ...  1.76916431e+00
  -1.01375378e+00  7.91025214e-01]]

[[ 1.46215957e+00  3.10605137e-03  1.02839177e+00 ...  1.62603366e+00
  -1.12411558e+00 -1.27840507e+00]
 [-3.06552093e+00 -2.21724106e+00  2.54365946e+00 ... -1.97930494e+00
  -9.91115793e-01 -5.00875141e+00]
 [ 2.18416996e+00  1.02708557e+00 -1.81769518e+00 ...  1.10008327e+00
   2.17065492e+00 -1.96419885e+00]
 ...
 [-6.97280143e-01 -2.10858465e+00  1.09087360e+00 ...  2.96835868e-01
  -1.93609031e-01 -2.01203452e+00]
 [-3.84798634e+00  1.10551338e+00  1.99044515e+00 ... -1.98141199e+00
  -1.18449026e+00 -3.62683418e+00]
 [ 6.77146968e-01  1.45158579e-01  1.49782246e-01 ... -1.23994092e-01
   1.22061707e+00 -2.91572033e+00]]

[[ 1.27462110e+00 -1.44172186e+00 -7.28790015e-01 ... -2.95924615e+00
   2.18614393e+00 -2.70474166e-01]
 [ 1.31324975e+00 -3.69330771e+00 -9.87616813e-01 ...  1.00305509e+00
  -1.39950441e+00  6.92987729e-01]
 [-1.89870599e+00  2.70334163e-01 -7.36285193e+00 ... -1.32358015e+00
  -1.24750571e+00 -1.47935628e+00]
 ...
 [ 3.50714374e+00 -3.85360440e+00 -2.48914799e+00 ... -1.41926617e+00
   1.90559011e+00 -7.50434415e-01]
 [-2.15001394e-01 -2.28129294e+00 -1.68668198e-01 ... -1.49844055e-01
  -7.99003149e-02 -1.86939408e+00]
 [-6.34239366e-01 -3.05927070e-01  1.58655486e+00 ...  5.07155771e-01
  -1.75746196e+00  1.73738370e+00]]]
ConvLayer : Input Matrix:
[[[2.78290873 3.32021319 3.36315089 ... 2.59298021 2.58794124 3.02355589]
  [2.96605718 4.76073164 3.36315089 ... 3.97280092 2.58794124 3.81508792]
  [2.96605718 4.76073164 3.07607369 ... 5.13549465 2.23876    3.39385797]
```

```
  ...
  [1.91510898 1.26677806 3.42663393 ... 3.05886408 0.62088239 3.21169731]
  [2.18226735 4.20127988 2.11363837 ... 3.49310888 1.0981733  2.31183329]
  [2.18226735 2.48238967 3.26128984 ... 1.98636755 3.45377324 3.96791644]]

 [[2.78290873 3.08969888 3.36315089 ... 1.92642974 2.58794124 0.        ]
  [3.64785195 3.08969888 3.36315089 ... 3.49001548 2.58794124 3.81508792]
  [0.82590827 1.19874099 2.61546433 ... 1.35548265 2.23876    2.08156444]
  ...
  [2.22005113 1.14098026 3.42663393 ... 2.41804181 1.96860525 2.70134368]
  [1.58775444 1.14098026 2.58431037 ... 3.1095558  2.01998749 2.31183329]
  [1.89994759 2.2564149  3.50962091 ... 2.98404859 3.05542696 3.96791644]]

 [[1.6933264  2.34474766 1.30454711 ... 3.11214408 3.60935775 1.27322504]
  [3.84340332 2.34474766 2.98147536 ... 3.49001548 3.60935775 3.29919974]
  [2.88944135 2.54473865 2.98147536 ... 2.21612345 3.49171722 3.29919974]
  ...
  [2.32350309 0.98522699 2.58431037 ... 4.02821098 3.13798906 0.26274194]
  [2.58780769 1.95177761 4.66816267 ... 3.1095558  2.01998749 1.3746442 ]
  [2.16957342 2.53737229 5.24787948 ... 3.34804501 2.01998749 1.9475122 ]]

 ...

 [[2.96516695 1.91381202 4.47356874 ... 1.81895141 1.75627792 3.04990496]
  [2.96516695 2.41609504 2.69385552 ... 2.18587254 3.47534531 1.97927511]
  [3.21588663 2.41609504 3.48218539 ... 2.75597906 2.12622854 1.39151476]
  ...
  [0.69252981 0.74047431 4.40765964 ... 4.6823111  2.88710826 0.89059921]
  [3.28934628 5.5459512  3.05277277 ... 2.71031893 2.88710826 1.37370447]
  [3.28934628 5.5459512  2.30411893 ... 3.69614538 1.99409336 5.05759839]]

 [[2.88190514 2.80823687 4.47356874 ... 0.67611203 2.19579089 3.30881092]
  [1.57587329 0.83147874 2.61757654 ... 2.18587254 2.4955374  2.62432965]
  [2.66143269 1.81191409 3.48218539 ... 2.18587254 2.4955374  2.06017249]
  ...
  [4.80424346 1.90040906 2.42658759 ... 4.37994421 1.85149034 4.29220511]
  [4.80424346 1.17626158 5.73147621 ... 3.81812132 3.21294946 0.89059921]
  [3.21848326 2.57566511 1.55415018 ... 3.69614538 1.99409336 5.05759839]]

 [[2.18416996 2.80823687 2.54365946 ... 1.62603366 2.18614393 1.82424301]
  [2.64282939 1.12190116 3.36052623 ... 1.91149695 4.95807127 2.62432965]
  [2.64282939 1.12190116 1.32620984 ... 1.8125462  4.95807127 2.08148838]
  ...
  [4.80424346 1.90040906 2.42658759 ... 3.81812132 2.5691984  4.29220511]
  [4.80424346 1.17626158 5.73147621 ... 3.81812132 3.21294946 2.88291599]
  [3.50714374 1.10551338 1.99044515 ... 2.50363526 1.90559011 2.30858144]]]
ConvLayer before ReLU output : [[[  59.52595394  -86.30723884 -130.35558592 ... -193.75145493
    -19.30158494   79.22969328]
  [  53.94219329  -88.53113403 -132.25460639 ... -165.5633177
      5.24959473  106.49153422]
  [  -9.51830701  -87.53570102 -111.49667812 ... -221.67208749
    -40.79342487   82.01231294]
  ...
  [  12.09445288  -74.30035464 -109.76514459 ... -201.57951121
    -50.06827452   84.89017518]
  [ -78.94015463 -114.36309347  -90.24534038 ... -200.05478852
    -18.95834922  127.44668443]
  [ -92.61263469  -82.0354807   -40.52920992 ... -164.50434355
     -0.80545664   53.13455367]]

 [[ 106.05087063 -101.02090791 -164.433521    ...  -60.91653487
    -41.08477583  122.76158386]
  [  86.54578572  -95.28485644 -184.97644893 ...   -2.83474811
     37.99826168  179.91017084]
  [ -18.69179581  -87.86489017 -137.86501601 ...  -93.70313961
```

```
  -30.84464657  142.52449673]
 ...
 [  10.92802221  -75.29768719 -160.57793932 ...  -83.41225432
   -44.23874142  136.15281037]
 [ -62.66129171 -103.09695164 -100.31652625 ... -138.10175138
   -38.9509096   182.86880374]
 [-117.46636722  -92.23726763    8.27781784 ... -143.49924643
    19.86730691   75.52123088]]

[[  64.24120435  -89.7613105  -242.25387281 ...  -98.02214595
   -17.47011706  155.08316735]
 [  74.16996688 -126.54201524 -259.63182587 ...  -89.46792158
    69.36497461  198.56683924]
 [ -17.98743625 -170.50527054 -225.7714901  ... -165.1855209
   -13.41387807  191.93646768]
 ...
 [ -15.9421937  -143.94151783 -251.16948367 ... -121.75189525
     1.6821769   193.33739479]
 [ -68.7359176  -227.13691332 -129.37789587 ... -200.03115179
    10.4575692   208.18036408]
 [ -87.35768282 -184.99102675    1.24506806 ... -193.39071868
    24.19444527   95.04185285]]

...

[[  54.54363743  -90.663666   -232.50177137 ...  -86.67313496
   -12.27209223  145.13384021]
 [  67.66783392 -114.98762152 -243.33914546 ...  -74.02368377
    70.77727608  180.02331694]
 [ -13.76152514 -148.16048073 -225.47158161 ... -125.02503141
   -10.14531483  169.28987868]
 ...
 [  -9.2165815  -153.29039205 -235.67048758 ... -133.44719718
     2.34944802  187.70905455]
 [ -61.42364659 -232.12612771 -126.33299837 ... -203.22866859
    20.76793656  210.22255679]
 [ -90.62861605 -192.78645104   -5.72154908 ... -197.46906294
    31.93457708  105.49111264]]

[[  81.34662235  -30.8168085  -236.47958367 ...  -10.63886238
   -37.51354767  170.93468206]
 [  89.03554286  -72.29033482 -254.05790337 ...   27.60116485
    11.14246545  222.84155373]
 [  22.22114913 -134.80470628 -215.20447276 ...  -28.17746238
   -84.37991342  199.85682097]
 ...
 [  26.97683786 -142.70200775 -231.40560344 ...  -38.83002872
   -86.41582972  212.65171767]
 [ -39.74541081 -215.90193699  -95.64369085 ...  -96.63238846
   -69.29038987  194.69415493]
 [ -63.6325626  -205.18726402    8.8251615  ... -122.94509899
   -50.61595262   72.55524407]]

[[  24.94820316  -63.37596769 -222.93229171 ...  -29.80247272
    26.67358473   82.48578514]
 [  53.31091273  -98.98851141 -220.20693992 ...   -5.02632081
    59.90212322  101.92931755]
 [  37.22975886 -126.7250839  -211.78851302 ...  -22.6442094
    -4.60970029  121.24182121]
 ...
 [  39.13387448 -130.97510427 -218.1470258  ...  -29.11346714
    -4.36071292  124.6424692 ]
 [  26.59085248 -192.90114792  -82.18392087 ... -112.22517579
    13.42298047  132.02697386]
 [ -12.02096065 -152.52603195   33.75356293 ...  -99.76547099
```

```
     -8.0501258    32.09146739]]]
ConvLayer : Input Matrix:
[[[106.05087063   0.           0.          ...  0.          69.36497461
   198.56683924]
  [  2.85811269   0.           0.          ...  0.           0.
   191.93646768]
  [  1.21119882   0.           0.          ...  0.           0.
   183.55258611]
  ...
  [  0.           0.           0.          ...  0.           0.
   186.57432698]
  [ 12.09445288   0.           0.          ...  0.           1.6821769
   193.33739479]
  [ 12.09445288   0.           8.27781784 ...  0.          24.19444527
   208.18036408]]

 [[ 83.96759095   0.           0.          ...  0.          83.36500385
   198.56683924]
  [  0.           0.           0.          ...  0.           1.63753007
   191.93646768]
  [  0.           0.           0.          ...  0.           0.
   191.61128068]
  ...
  [  0.           0.           0.          ...  0.           2.97149972
   191.05703452]
  [  0.           0.           0.          ...  0.           1.6821769
   195.89248465]
  [  0.           0.           1.24506806 ...  0.          29.57958583
   208.18036408]]

 [[ 79.49600376   0.           0.          ...  0.          83.36500385
   191.38800477]
  [  0.           0.           0.          ...  0.           1.63753007
   191.38800477]
  [  0.           0.           0.          ...  0.           1.55410145
   191.61128068]
  ...
  [  0.           0.           0.          ...  0.           1.08981429
   192.19999469]
  [  0.           0.           0.          ...  0.           5.48807098
   195.75877664]
  [  0.           0.           0.          ...  0.          29.57958583
   196.54047336]]

 ...

 [[ 81.55885166   0.           0.          ...  0.          82.96770226
   192.56082135]
  [  0.           0.           0.          ...  0.           0.
   196.18541988]
  [  0.           0.           0.          ...  0.           0.
   196.18541988]
  ...
  [  0.29943605   0.           0.          ...  0.           1.02401798
   201.48276074]
  [  0.           0.           0.          ...  0.           0.
   197.50254985]
  [  0.           0.           0.          ...  0.          27.63557233
   201.59074034]]

 [[ 77.31110443   0.           0.          ...  0.          82.96770226
   190.90865849]
  [  0.           0.           0.          ...  0.           1.9154658
   196.18541988]
  [  0.           0.           0.          ...  0.           5.372042
```

```
   196.18541988]
  ...
  [  0.           0.           0.         ...   0.           1.68981423
   193.34021196]
  [  0.           0.           0.         ...   0.           2.34944802
   191.9080408 ]
  [  0.           0.           0.         ...   0.          31.93457708
   210.22255679]]

 [[ 89.03554286   0.           0.         ...  27.60116485  70.77727608
   222.84155373]
  [ 37.22975886   0.           0.         ...   0.           3.16529073
   210.65592745]
  [ 44.21325605   0.           0.         ...   0.           2.39856626
   212.18660769]
  ...
  [ 35.89165713   0.           0.         ...   0.           3.94681128
   204.90032844]
  [ 39.13387448   0.           0.         ...   0.           3.94681128
   217.45317028]
  [ 39.13387448   0.          33.75356293 ...   0.          31.93457708
   212.65171767]]]
ConvLayer before ReLU output : [[[   653.25361784   -862.59591369  -2740.4794099  ...   4017.20095652
    -3576.76759586   4717.20590601]
  [   317.58119497  -3266.27837709 -11051.11808865 ...   -295.61137877
    -7192.0426895    1824.424606  ]
  [   345.63430886  -2800.35059846  -9468.62243877 ...    142.34939833
    -6465.82809276   2142.32118392]
  ...
  [   342.27713151  -2758.34301481  -9387.60539106 ...     36.21007145
    -6442.05011147   2175.92641483]
  [   271.20609533  -2860.85148314  -9493.46660797 ...   -777.39480588
    -7270.71358503   2231.35663245]
  [   862.97043499  -2263.48330076  -8827.48944039 ...   6097.180241
    -2099.42159483   2731.51465773]]

 [[   126.79427599   2108.57734419  -2703.74236993 ...   8086.12071925
    -8638.2534312    2510.22528014]
  [  1179.75290352   3082.06804891 -11675.15302264 ...   1160.56448246
   -11660.24267306   3693.89619146]
  [   987.27193705   2899.12229312  -9829.11926955 ...   1951.29006582
   -10833.73425362   3385.08676885]
  ...
  [   949.78220175   2973.77463206  -9693.85135829 ...   2035.51402444
   -10910.26688301   3455.60360858]
  [   802.67734659   2993.22073779  -9643.97633758 ...   2056.35480577
   -11901.71323964   3173.67596823]
  [  1767.58787303   2636.53998976 -10515.23332963 ...   3719.26348699
    -6186.42977446   4945.29452037]]

 [[   214.70043834   1644.7404711   -2687.57438516 ...   7307.79663323
    -7735.92019103   2769.82530713]
  [  1026.92193573   2136.80914941 -11326.31774345 ...    881.9545057
   -10695.53242324   3425.97205556]
  [   826.05514147   1973.78724367  -9496.79487897 ...   1668.78759524
    -9921.35845176   3122.96927968]
  ...
  [   788.2535857    1844.29065039  -9500.12449363 ...   1848.43696638
   -10069.51563608   3108.27590889]
  [   695.12772506   1932.91801492  -9512.36326538 ...   1721.60936712
   -10963.66919479   2922.72850992]
  [  1505.71558569   1814.92471884 -10038.27204118 ...   3939.67721868
    -5472.92778725   4437.40366438]]

 ...
```

```

 [[   237.05053092   1605.27864802   -2724.13632851 ...   7226.59519271
    -7817.35110651   2810.16307843]
  [  1023.28279414   2094.8341213   -11438.47511005 ...    969.12459674
   -10817.86524941   3500.99319829]
  [   793.33470236   1946.58281372   -9710.43330871 ...   1852.98271959
   -10242.45586194   3119.20230011]
  ...
  [   843.00824605   1977.77043682   -9527.99471386 ...   1810.56524875
   -10140.3020951    3245.81868712]
  [   745.45220945   1962.48833751   -9740.56178505 ...   1576.27776211
   -11107.86401938   3097.77095646]
  [  1507.78901708   1785.00174274 -10235.46456123 ...   4104.13067588
    -5458.8344707    4571.128427  ]]

 [[    92.46101633   1637.09545867   -3142.67849032 ...   7454.63484736
    -8319.78814745   3230.9566687 ]
  [   962.92615006   1878.86921977 -12261.24271114 ...   1245.03913558
   -11796.8595042    3891.92930982]
  [   731.00527345   1614.98792128 -10827.25296136 ...   2177.44768347
   -11222.32375419   3584.03963719]
  ...
  [   756.04563391   1674.55893435 -10359.91566522 ...   2165.20373444
   -11132.66422047   3372.42556898]
  [   707.35831358   1716.53749296 -10689.34160593 ...   1843.7494964
   -12281.14459766   3412.22386964]
  [  1216.15079111   1735.79162957 -11069.41341701 ...   4935.87584572
    -5944.07700487   4975.55710105]]

 [[  1212.25237932   1345.22892338      17.47388627 ...   4156.04508119
    -5860.67868968   -581.92598251]
  [  1471.96745834   3800.53087149   -6198.56076478 ...   -846.76385715
    -5754.11382867   2527.71717332]
  [  1403.53762751   3309.69802591   -5393.29246299 ...   -144.79018829
    -5656.26654214   2032.92896365]
  ...
  [  1278.19601653   3259.93354576   -5129.89526908 ...    -50.46105793
    -5615.95812639   1832.01217725]
  [  1061.88914522   3511.97754636   -4964.49850609 ...   -251.55106671
    -6242.37159969   1678.30625299]
  [  3279.13375447   2421.00465384   -6914.58670757 ...   1113.65288599
    -3156.25865522   3616.87717427]]]
ConvLayer : Input Matrix:
[[[ 653.25361784    0.            0.          ... 4017.20095652
      0.         4717.20590601]
  [ 317.58119497    0.            0.          ...    0.
      0.         1824.424606  ]
  [ 345.63430886    0.            0.          ...  142.34939833
      0.         2142.32118392]
  ...
  [ 342.27713151    0.            0.          ...   36.21007145
      0.         2175.92641483]
  [ 271.20609533    0.            0.          ...    0.
      0.         2231.35663245]
  [ 862.97043499    0.            0.          ... 6097.180241
      0.         2731.51465773]]

 [[ 126.79427599 2108.57734419    0.          ... 8086.12071925
      0.         2510.22528014]
  [1179.75290352 3082.06804891    0.          ... 1160.56448246
      0.         3693.89619146]
  [ 987.27193705 2899.12229312    0.          ... 1951.29006582
      0.         3385.08676885]
  ...
  [ 949.78220175 2973.77463206    0.          ... 2035.51402444
```

```
      0.         3455.60360858]
  [ 802.67734659 2993.22073779    0.         ... 2056.35480577
      0.         3173.67596823]
  [1767.58787303 2636.53998976    0.         ... 3719.26348699
      0.         4945.29452037]]

 [[ 214.70043834 1644.7404711     0.         ... 7307.79663323
      0.         2769.82530713]
  [1026.92193573 2136.80914941    0.         ...  881.9545057
      0.         3425.97205556]
  [ 826.05514147 1973.78724367    0.         ... 1668.78759524
      0.         3122.96927968]
  ...
  [ 788.2535857  1844.29065039    0.         ... 1848.43696638
      0.         3108.27590889]
  [ 695.12772506 1932.91801492    0.         ... 1721.60936712
      0.         2922.72850992]
  [1505.71558569 1814.92471884    0.         ... 3939.67721868
      0.         4437.40366438]]

 ...

 [[ 237.05053092 1605.27864802    0.         ... 7226.59519271
      0.         2810.16307843]
  [1023.28279414 2094.8341213     0.         ...  969.12459674
      0.         3500.99319829]
  [ 793.33470236 1946.58281372    0.         ... 1852.98271959
      0.         3119.20230011]
  ...
  [ 843.00824605 1977.77043682    0.         ... 1810.56524875
      0.         3245.81868712]
  [ 745.45220945 1962.48833751    0.         ... 1576.27776211
      0.         3097.77095646]
  [1507.78901708 1785.00174274    0.         ... 4104.13067588
      0.         4571.128427  ]]

 [[  92.46101633 1637.09545867    0.         ... 7454.63484736
      0.         3230.9566687 ]
  [ 962.92615006 1878.86921977    0.         ... 1245.03913558
      0.         3891.92930982]
  [ 731.00527345 1614.98792128    0.         ... 2177.44768347
      0.         3584.03963719]
  ...
  [ 756.04563391 1674.55893435    0.         ... 2165.20373444
      0.         3372.42556898]
  [ 707.35831358 1716.53749296    0.         ... 1843.7494964
      0.         3412.22386964]
  [1216.15079111 1735.79162957    0.         ... 4935.87584572
      0.         4975.55710105]]

 [[1212.25237932 1345.22892338   17.47388627 ... 4156.04508119
      0.            0.        ]
  [1471.96745834 3800.53087149    0.         ...    0.
      0.         2527.71717332]
  [1403.53762751 3309.69802591    0.         ...    0.
      0.         2032.92896365]
  ...
  [1278.19601653 3259.93354576    0.         ...    0.
      0.         1832.01217725]
  [1061.88914522 3511.97754636    0.         ...    0.
      0.         1678.30625299]
  [3279.13375447 2421.00465384    0.         ... 1113.65288599
      0.         3616.87717427]]]
ConvLayer before ReLU output : [[[  58098.68404222 -193471.91071446  142547.31948476 ...
    -64354.96568824   55502.95214574  -12066.64729771]
```

```
  [  49933.69644814 -213548.1993692   225114.67455546 ...
     49340.31719641  -68523.40654342   56406.82264333]
  [  37101.66194265 -211939.66243034  211398.58398857 ...
     79540.5937674  -100832.86288535   79480.08661192]
  ...
  [  38609.18543326 -221708.72077544  203809.48736017 ...
     65778.76105065  -86409.86841292   74533.60096222]
  [  45514.19539715 -201159.27454479  235209.84038107 ...
     68817.82563157  -85657.56251745   64399.91045612]
  [  43453.98122654  -72025.1541494   244536.88562771 ...
    103299.36421869 -117793.3532441    45460.1392419 ]]

 [[ 164314.39526921 -336963.4061955   165365.8435381  ...
    -22134.02747786   71768.92781698  -21685.43239892]
  [ 146665.28812861 -345484.49094101  275396.67999751 ...
    147902.43507827 -126307.53313959   22691.84438199]
  [ 130326.7366419  -333427.88133087  258733.69513998 ...
    187868.39447325 -175228.45157927   48262.05327785]
  ...
  [ 140435.57634275 -352024.06476506  247839.41254637 ...
    169807.41815979 -151859.45428334   46579.1652161 ]
  [ 136332.95973369 -318910.47658294  284612.73348378 ...
    168983.99515387 -154244.25529804   24355.11797828]
  [  48012.75537951 -109636.21349388  310215.39447639 ...
    187169.14602133 -202344.90537327  -15563.6942426 ]]

 [[ 189006.00671447 -351675.15053287  154900.47472727 ...
    -20960.00220651   74374.36129957  -31551.42810917]
  [ 163828.44649669 -360083.10576315  262892.00912322 ...
    149149.37550753 -127400.83948938    3144.84927399]
  [ 145953.33177197 -346833.24213119  244973.376913   ...
    188962.24940265 -177729.80461557   28352.72595861]
  ...
  [ 157819.95159154 -367639.30544585  235142.48335903 ...
    170639.4613495  -154083.77279482   27764.41197236]
  [ 144238.7026677  -330362.0195781   276180.13019393 ...
    170613.26424009 -156972.04494221    4052.85120194]
  [  38849.46687466 -114338.10712484  306657.56170975 ...
    189363.76387866 -205919.67727967  -30338.94478941]]

 ...

 [[ 184595.57096823 -348660.5780833   157218.25403094 ...
    -30348.81935877   76303.37229122  -32936.02136103]
  [ 160357.01595332 -363928.81716144  265651.28906051 ...
    138401.80228037 -123092.1674589     7031.35727107]
  [ 145295.62909782 -356845.95331693  251398.70105006 ...
    179694.07112817 -173325.94709909   32902.17552455]
  ...
  [ 152090.36943668 -367970.09555338  236671.04382206 ...
    160821.29493632 -150951.69347586   30320.12307875]
  [ 140016.71413118 -330008.86755935  284658.21842942 ...
    161124.98153225 -153436.73726031    6868.88687859]
  [  38552.87861632 -115803.88566821  313799.2521172  ...
    186382.79972852 -203982.9040739   -22493.84197779]]

 [[ 173162.51447106 -346134.75682317  152591.95980378 ...
    -10999.39845258   68628.18267157  -18579.08779412]
  [ 161325.09540891 -356153.05655067  271094.66744046 ...
    155357.32187921 -126690.14739914   13973.91201051]
  [ 145627.09295741 -348625.71252812  259108.01995207 ...
    197421.75425859 -179166.96363844   40460.81938644]
  ...
  [ 151704.01699111 -361612.62330951  243601.26858398 ...
    180152.95382215 -154436.32527742   40225.29403833]
```

```
  [ 144314.90382911 -327810.77915869  278164.32163359 ...
    174833.9540867  -159827.52135812   13916.83143564]
  [  50960.52143198 -110114.52458336  319764.11099609 ...
    191763.933414   -207540.4776138   -26127.87730775]]

 [[ 117709.28872071 -269102.4314352   115437.01691988 ...
     81720.56206206   30090.69289705   30001.8352346 ]
  [ 138212.55859015 -252407.78314574  197874.26950932 ...
    213496.5217991  -125242.39178497   28014.05420943]
  [ 140127.24914236 -237563.36247923  204028.86949378 ...
    247217.96719142 -165919.46384084   39585.0306973 ]
  ...
  [ 143434.20022755 -249003.17103037  192236.80754265 ...
    233126.22062541 -144959.41333781   43262.95649153]
  [ 127459.25358258 -229184.93263032  209755.94700732 ...
    229548.12730425 -151644.7838524    24001.78660614]
  [  63270.12726242  -55868.60513076  188787.05132474 ...
    186549.49351362 -193138.39713141  -37353.46174205]]]
ConvLayer : Input Matrix:
[[[ 58098.68404222      0.          142547.31948476 ...      0.
    55502.95214574      0.        ]
  [ 49933.69644814      0.          225114.67455546 ...  49340.31719641
        0.          56406.82264333]
  [ 37101.66194265      0.          211398.58398857 ...  79540.5937674
        0.          79480.08661192]
  ...
  [ 38609.18543326      0.          203809.48736017 ...  65778.76105065
        0.          74533.60096222]
  [ 45514.19539715      0.          235209.84038107 ...  68817.82563157
        0.          64399.91045612]
  [ 43453.98122654      0.          244536.88562771 ... 103299.36421869
        0.          45460.1392419 ]]

 [[164314.39526921      0.          165365.8435381  ...      0.
    71768.92781698      0.        ]
  [146665.28812861      0.          275396.67999751 ... 147902.43507827
        0.          22691.84438199]
  [130326.7366419       0.          258733.69513998 ... 187868.39447325
        0.          48262.05327785]
  ...
  [140435.57634275      0.          247839.41254637 ... 169807.41815979
        0.          46579.1652161 ]
  [136332.95973369      0.          284612.73348378 ... 168983.99515387
        0.          24355.11797828]
  [ 48012.75537951      0.          310215.39447639 ... 187169.14602133
        0.              0.        ]]

 [[189006.00671447      0.          154900.47472727 ...      0.
    74374.36129957      0.        ]
  [163828.44649669      0.          262892.00912322 ... 149149.37550753
        0.           3144.84927399]
  [145953.33177197      0.          244973.376913   ... 188962.24940265
        0.          28352.72595861]
  ...
  [157819.95159154      0.          235142.48335903 ... 170639.4613495
        0.          27764.41197236]
  [144238.7026677       0.          276180.13019393 ... 170613.26424009
        0.           4052.85120194]
  [ 38849.46687466      0.          306657.56170975 ... 189363.76387866
        0.              0.        ]]

 ...

 [[184595.57096823      0.          157218.25403094 ...      0.
    76303.37229122      0.        ]
```

```
  [160357.01595332      0.          265651.28906051 ... 138401.80228037
        0.             7031.35727107]
  [145295.62909782      0.          251398.70105006 ... 179694.07112817
        0.            32902.17552455]
  ...
  [152090.36943668      0.          236671.04382206 ... 160821.29493632
        0.            30320.12307875]
  [140016.71413118      0.          284658.21842942 ... 161124.98153225
        0.             6868.88687859]
  [ 38552.87861632      0.          313799.2521172  ... 186382.79972852
        0.                0.        ]]

 [[173162.51447106      0.          152591.95980378 ...      0.
    68628.18267157      0.        ]
  [161325.09540891      0.          271094.66744046 ... 155357.32187921
        0.            13973.91201051]
  [145627.09295741      0.          259108.01995207 ... 197421.75425859
        0.            40460.81938644]
  ...
  [151704.01699111      0.          243601.26858398 ... 180152.95382215
        0.            40225.29403833]
  [144314.90382911      0.          278164.32163359 ... 174833.9540867
        0.            13916.83143564]
  [ 50960.52143198      0.          319764.11099609 ... 191763.933414
        0.                0.        ]]

 [[117709.28872071      0.          115437.01691988 ...  81720.56206206
    30090.69289705  30001.8352346 ]
  [138212.55859015      0.          197874.26950932 ... 213496.5217991
        0.            28014.05420943]
  [140127.24914236      0.          204028.86949378 ... 247217.96719142
        0.            39585.0306973 ]
  ...
  [143434.20022755      0.          192236.80754265 ... 233126.22062541
        0.            43262.95649153]
  [127459.25358258      0.          209755.94700732 ... 229548.12730425
        0.            24001.78660614]
  [ 63270.12726242      0.          188787.05132474 ... 186549.49351362
        0.                0.        ]]]
ConvLayer before ReLU output : [[[ 4545793.83679853  6119166.95612676  6178484.85073528 ...
    2134719.19798516   -63921.84659327 -1337211.58041342]
  [15442335.98913171  6532265.1219183   4241942.42160134 ...
     949832.90315975  4361115.03928851  2682707.19981612]
  [17386719.70511443  6373756.34655366  3669525.51710732 ...
     755325.13591153  5170868.26322278  3335294.69316039]
  ...
  [17338563.6947727   6396380.02201548  3707545.60662673 ...
     766857.84710187  5133761.76454293  3357598.78094414]
  [17469899.25825548  5644677.63428755  2853038.13584494 ...
     644206.95985711  5412564.91985235  3481209.70865648]
  [16270349.06008183  1408716.8137525  -1564640.84264143 ...
     306751.00497729  6020590.02500007  3189616.01071096]]

 [[ 7518456.95275321  7406856.26023747  7108788.20459085 ...
   -1232428.37717295 -4695662.6009614  -1420786.8759428 ]
  [13405263.14818262  8603947.82624617  5280523.88524402 ...
   -5513334.79124475  -331776.18573134  2600490.31577874]
  [14301468.28475474  8596779.94816421  4766183.29793813 ...
   -6169992.96984055   630881.8820173   3289051.26748164]
  ...
  [14422063.29624643  8601525.46668501  4706769.00369852 ...
   -6212971.5568458    568180.41072025  3321300.26728836]
  [14040296.85159239  7872341.5294383   4095914.82628745 ...
   -6068045.64817665  1531159.28018916  3632163.09677273]
  [11277119.58091383  3499397.98233268  1021252.40744377 ...
```

```
   -4162176.21325016  5935469.31856704  4390288.2495808 ]]

 [[ 7889292.64551156  7465723.95659528  7227480.4883013  ...
   -1800382.49677908 -5490892.63482856 -1332432.39305813]
  [12592083.08070556  8778137.98883679  5473109.97404682 ...
   -6587515.46897777 -1227289.17223259  2532882.81591277]
  [13394720.71717763  8827759.08101651  5091205.13563131 ...
   -7277433.45673336  -172234.39091524  3260338.63779996]
  ...
  [13617492.24094965  8850176.08367058  4971438.89049927 ...
   -7350269.36313838  -169166.05624705  3325830.71773923]
  [13090106.06227689  8099765.47594873  4411413.22883529 ...
   -7119969.92580282   862307.08401694  3639667.90195754]
  [10073638.34309632  3858132.16399913  1692852.80874791 ...
   -4862458.34819983  5984175.75405487  4574703.95012976]]

 ...

 [[ 7989941.8998917   7604975.01405309  7297279.31408973 ...
   -1832774.64288986 -5536050.0426882  -1387589.34434469]
  [12933908.32228405  8997378.51853662  5522405.86293813 ...
   -6699949.14793317 -1338699.51388313  2509657.90367355]
  [13889715.17911603  9150645.47748271  5184616.75836795 ...
   -7480009.43826063  -360953.51768652  3266074.91285376]
  ...
  [13598999.89638391  8998576.09823901  5085764.69376523 ...
   -7411261.99021791  -174474.61484937  3292710.84386918]
  [13276497.73159196  8253659.20813499  4419110.25470916 ...
   -7192423.84454647   865659.18193331  3611544.48604063]
  [10423440.22691292  3880012.94635373  1556051.1256114  ...
   -4899091.86711492  5988795.10401306  4613039.82590934]]

 [[ 7367201.24536233  6873041.94501115  7214858.08939702 ...
   -1654271.25792036 -5545040.87599066  -928459.60307323]
  [11213450.01570631  8267899.09191869  6120161.61028389 ...
   -6243876.81024488 -1280764.3763586   2468207.13045223]
  [12062356.29760192  8456278.61162099  5908510.81781316 ...
   -6993473.38655863  -234278.85416776  3174982.27741888]
  ...
  [11831935.4738793   8287901.69596612  5755563.18546423 ...
   -6932154.77531226  -156198.48327673  3189648.24009875]
  [11470558.93569036  7508202.83628194  5129844.9796914  ...
   -6680016.87810278   893265.71707948  3518776.49563371]
  [ 8602628.43148693  3913042.41950887  2836493.47589101 ...
   -4693088.2427039   6451911.14444801  4563431.25645836]]

 [[ 3548527.27862795  2461623.09406749  6243185.66872721 ...
    -212973.48631388 -4906706.77260253  1476635.51492342]
  [ 1975832.1035378   3572716.81353391  8715613.35379806 ...
   -2861541.22799593  -411425.02943386  2342093.44077675]
  [ 1727569.32292268  3853459.57215342  9379822.81404984 ...
   -3296549.88540929   816495.55633326  2609521.52715301]
  ...
  [ 1679753.11786453  3785091.37991508  9200778.47739733 ...
   -3288368.00964036   892019.01009304  2560076.51069125]
  [ 1339535.63697071  3760659.37107209  8994313.94058043 ...
   -3424448.78156352  2134804.29211949  2600235.71426498]
  [ -430988.37389859  3493024.24830385  8207297.91182273 ...
   -2899610.73377349  7663249.93295785  3047503.83326688]]]
Output vector: [      0.                0.          1840024.67284437       0.
       0.                0.                0.          8137793.07131413
  353828.86617409 5303816.24340175]

Process finished with exit code 0
```

Listing 4.2: AlexNet architecture