**Task**

Convert a gray-scale image \( P \) (loaded as initial state) into a binary image where each pixel \( p_{ij} \in P \) is converted into “black” (“red” in pseudo-color) if, and only if, \( p_{ij} \) has a gray scale intensity exceeding a prescribed value equal to the CNN threshold \( z^* \). In the two extreme cases when \( z^* = 1 \) and \( z^* = -1 \) all output pixels will be printed in black (red), or in white (blue) respectively.

**Prescription:**

**Cloning Template**

\[
\begin{array}{c|ccc|c}
\hline
z : & 0 & 0 & 0 & z^* \\
\hline
B : & 0 & 0 & 0 & 0 \\
A : & 0 & 2 & 0 & 0 \\
\hline
\end{array}
\]

**Boundary Condition**

Fixed: \( x_{i^*j^*} = 0, \ u_{i^*j^*} = 0 \)

(i\(^*\)j\(^*\) denotes boundary cells)

**Initial State**

\( x_{ij}(0) = \) a given gray-scale image

**Example 1:** Array Size = 30 x 48; \( z^* = -0.5 \)

**Input Image:** arbitrary

**Initial State**

**Output Image**

**Example 2:** Array Size = 256 x 320; \( z^* = -0.5 \)
### Name:
**HORIZONTAL TRANSLATION CNN**

### Task
Shift a binary image by one-pixel to the right. Rightmost pixels of all image rows are discarded whereas leftmost pixels of all rows are filled with the background color. For left translation, simply rotate B by 180° about the center.

### Cloning Template

<table>
<thead>
<tr>
<th>z:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>B:</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>1 0 0</td>
</tr>
<tr>
<td></td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

| A: | 0 0 0 |
|    | 0 1 0 |
|    | 0 0 0 |

### Boundary Condition

**Fixed:** \( x_{i^*j^*} = 0, \ u_{i^*j^*} = -1 \)

(i\(^*\)j\(^*\) denotes boundary cells)

### Initial State

\( x_{ij}(0) = 0 \) (white in pseudo-color)

### Example 1: Array Size = 16 x 16

- **Input Image**
- **Initial State**
- **Output Image**

### Example 2: Array Size = 64 x 64
**Name:** VERTICAL TRANSLATION CNN

**Task**
Shift a binary image by one-pixel to the top. Uppermost pixels of all image columns are discarded whereas lowermost pixels of all columns are filled with the background color. For a downward translation, simply rotate B by 180° about the center.

**Prescription:**

**Cloning Template**

<table>
<thead>
<tr>
<th>z</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>0 0 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td></td>
<td>0 1 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

**Boundary Condition**

Fixed: \( x_{i*j*} = 0, \ u_{i*j*} = -1 \)

(i*j* denotes boundary cells)

**Initial State**

\( x_{ij}(0) = 0 \) (white in pseudo-color)

**Example 1:** Array Size = 16 x 16

- **Input Image**
- **Initial State**
- **Output Image**

**Example 2:** Array Size = 64 x 64
Shift a binary image by one-pixel in a right-down diagonal direction. This operation can be seen as a combination of a horizontal and a vertical translation. For left-up translation, simply rotate B by 180° about the center.

### Cloning Template

<table>
<thead>
<tr>
<th>z</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 1 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

### Boundary Condition

**Fixed:** $x_{i,j}^* = 0$, $u_{i,j}^* = -1$

(i*j* denotes boundary cells)

### Initial State

$x_{ij}(0) = 0$ (white in pseudo-color)

### Example 1: Array Size = 16 x 16

- **Input Image**
- **Initial State**
- **Output Image**

### Example 2: Array Size = 64 x 64
**Name:** POINT EXTRACTION CNN

**Task Prescription:** Extract all isolated black pixels (red in pseudo-color) from a binary image. The task is complement (opposite) to that performed by a Point Removal CNN.

<table>
<thead>
<tr>
<th>Cloning Template</th>
<th>Boundary Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( z : -8 )</td>
<td>Fixed: ( x_{i^<em>j^</em>} = -1, \ u_{i^<em>j^</em>} = -1 )</td>
</tr>
<tr>
<td>( B : )</td>
<td>( (i^<em>j^</em> \text{ denotes boundary cells}) )</td>
</tr>
<tr>
<td>( -1 -1 -1 )</td>
<td>Initial State</td>
</tr>
<tr>
<td>( -1 1 -1 )</td>
<td>( x_{ij}(0) = 0 ) (white in pseudo-color)</td>
</tr>
<tr>
<td>( -1 -1 -1 )</td>
<td></td>
</tr>
<tr>
<td>( A : )</td>
<td></td>
</tr>
<tr>
<td>( 0 0 0 )</td>
<td></td>
</tr>
<tr>
<td>( 0 1 0 )</td>
<td></td>
</tr>
<tr>
<td>( 0 0 0 )</td>
<td></td>
</tr>
</tbody>
</table>

**Example 1:** Array Size = 30 x 48

**Input Image**

**Initial State**

**Output Image**

**Example 2:** Array Size = 260 x 280
**Name:** POINT REMOVAL CNN

**Task Prescription:** Delete all isolated black (red in pseudo-color) pixels in a binary image. The task is complement (opposite) of that performed by a Point Detection CNN.

### Cloning Template

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Boundary Condition

Fixed: \( x_{i,j} = -1, \  u_{i,j} = -1 \)

\( (i,j) \) denotes boundary cells

### Initial State

\( x_{ij}(0) = 0 \) (white in pseudo-color)

### Example 1: Array Size = 30 x 48

#### Input Image

![Input Image](image1.png)

#### Initial State

![Initial State](image2.png)

#### Output Image

![Output Image](image3.png)

### Example 2: Array Size = 260 x 280

#### Input Image

![Input Image](image4.png)

#### Initial State

![Initial State](image5.png)

#### Output Image

![Output Image](image6.png)
**Name:** LOGIC “NOT” OPERATION CNN

**Task:** Perform a logic NOT operation, or set complement $\overline{U}$ on elements of a binary image $U$ which is loaded as CNN input. Pixels initially black (red in pseudo-color) become white (blue in pseudo-color), and vice versa.

<table>
<thead>
<tr>
<th>Cloning Template</th>
<th>Boundary Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>z:</strong> 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td><strong>B:</strong> 0 -2 0</td>
<td>0 1 0</td>
</tr>
</tbody>
</table>

**Example 1:** Array Size = 30 x 48

- **Input Image**
- **Initial State**
- **Output Image**

**Example 2:** Array Size = 256 x 300

**Initial State**

- $x_{i,j} = 0$, $u_{i,j} = 0$
- ($i,j$ denotes boundary cells)

- $x_{i,j}(0)$ = arbitrary, e.g. $x_{i,j}(0) = 0$
- (white in pseudo-color)
**Task Prescription:**
Perform a pixel-wise logic AND operation ($\land$), or set intersection ($\cap$), between two binary images loaded as CNN initial state and CNN input.

**Cloning Template**

<table>
<thead>
<tr>
<th>z</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>0 1 0</td>
<td>0 2 0</td>
</tr>
<tr>
<td></td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

**Boundary Condition**

Fixed: $x_{ij^*} = 0$, $u_{ij^*} = 0$
($i^*j^*$ denotes boundary cells)

**Initial State**

$x_i(0) =$ another binary image

**Example 1:** Array Size = 30 x 48

- **Input Image**
- **Initial State**
- **Output Image**

**Example 2:** Array Size = 176 x 176
Perform a pixel-wise logic OR operation (\( \lor \)), or set union (\( \cup \)), between two binary images loaded as CNN initial state and CNN input.

Cloning Template

<table>
<thead>
<tr>
<th>z</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0</td>
<td>0 2 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td></td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

Boundary Condition

Fixed: \( x_{i^*j^*} = 0, \ u_{i^*j^*} = 0 \)  
(i* denotes boundary cells)

Initial State

\( x_{ij}(0) = \) another binary image

Example 1: Array Size = 30 x 48

Input Image

Initial State

Output Image

Example 2: Array Size = 256 x 256
**Name:** EDGE DETECTION CNN

*Task Prescription:* Extract edges of objects in a binary image where each black (red in pseudo-color) pixel with at least one white (blue in pseudo-color) nearest neighbor is defined to be an edge cell.

**Cloning Template**

<table>
<thead>
<tr>
<th>z</th>
<th>-0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Boundary Condition**

Fixed: $x_{i\ast j\ast} = 0$, $u_{i\ast j\ast} = 0$

($i\ast j\ast$ denotes boundary cells)

**Initial State**

$x_{ij}(0) = 0$ (white in pseudo-color)

**Example 1:** Array Size = 44 x 44

- **Input Image**
- **Initial State**
- **Output Image**

**Example 2:** Array Size = 140 x 140

- **Input Image**
- **Initial State**
- **Output Image**