### Task Prescription:

Extract contours which resemble edges (resulting from big changes in gray level intensities) from gray-scale images.

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

| 0.7 | a | a | a | a | 0 | a | a | a | a | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |

### Example 2: Array Size = 256 x 256

\[
a = 0.5 \left[ \text{sgn}(u_{ij} - u_{kl} - 0.18) - \text{sgn}(u_{ij} - u_{kl} + 0.18) + 1 \right]
\]

#### Boundary Condition

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

(i*j* denotes boundary cells)

#### Initial State

\[ x_{ij}(0) = u_{ij} = \text{input image} \]

### Input Image

![Input Image](image1)

### Initial State

![Initial State](image2)

### Output Image

![Output Image](image3)
GRADIENT DETECTION CNN

Task
Detect all locations (coded in black, red in pseudo-color) in a gray-scale image (representing some field intensity) where the gradient of the field is smaller than some prescribed threshold $z^*$. 

**Name:** task

**Prescription:**

- **Boundary Condition**
  - Fixed: $x_{i*j*} = 0$, $u_{i*j*} = 0$
  - (i*j* denotes boundary cells)

- **Initial State**
  - $x_{ij}(0) = an arbitrary image$

**Example 1:** Array Size = 30 x 48; $z^* = -6$

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

**Example 2:** Array Size = 320 x 176; $z^* = -6$

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

**G**

$z^*$ | a | a | a | a | 0 | a | a | a | a | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0

$a = -10|\Delta u| + 5|\Delta u + 0.2| + 5|\Delta u - 0.2| + 0.5\text{sgn}(\Delta u + 0.2) - 0.5\text{sgn}(\Delta u - 0.2) - 2$

where: $\Delta u = u_{ij} - u_{kl}$
Name: MOTION DETECTION CNN

Task Prescription: Detect objects moving horizontally to the right with a speed of 1 pixel/delay time.

**Cloning Template**

<table>
<thead>
<tr>
<th>G:</th>
<th>-2</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1.5</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>-0.1</th>
<th>-0.1</th>
<th>-0.1</th>
<th>-0.1</th>
<th>-0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_T:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Delay Time** \( T = 10 \)

Example: Array Size = 20 x 20

**snapshots of the input stream**

- \( t = 0 \)
- \( t = 3 \, T \)
- \( t = 1 \, T \)
- \( t = 4 \, T \)

**corresponding outputs**
**Name:** IMAGE DIFFERENCE COMPUTATION CNN

**Task Prescription:**
Take the logic difference of the delayed and current input image in continuous motion together with noise filtering. Only the moving parts of an image are extracted.

### Cloning Template

<table>
<thead>
<tr>
<th>G:</th>
<th>-4.75</th>
<th>0.25</th>
<th>0.25</th>
<th>0.25</th>
<th>2</th>
<th>0.25</th>
<th>0.25</th>
<th>0.25</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>G_\text{T}:</td>
<td>0</td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.25</td>
<td>-2</td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Delay Time

T = 10

### Example:

**Array Size = 44 x 44**

- u_ij(0)
- u_ij(10T)

*two consecutive steps from t to t + 10T of the input stream*

*output (10T)*
SPEED DETECTION CNN

Task Prescription: Detect objects moving with a speed slower than 1 pixel/delay time in arbitrary directions.

Cloning Template

**G:**

-2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**G_t:**

0 0 0 0 0 0 0 0 0 0.68 0.68 0.68 0.68 0.68 0.68 0.68 0.68

Delay Time: **T = 10**

Example: Array Size = 16 x 16

```
t = 0  t = 10 T  t = 20 T  t = 30 T  t = 40 T
```

five consecutive steps of the input stream (10 T)

```
t = 0  t = 10 T  t = 20 T  t = 30 T  t = 40 T
```

corresponding outputs
Name: HERRING-GRID ILLUSION CNN

Simulate the well-known illusion where an input image of closely-spaced uniform black squares is perceived by humans at a distance as if there were gray patches present at the intersections of the horizontal and vertical white grid lines separating the squares. The CNN output must generate these gray patches and superimpose them at corresponding locations on the input image.

**Cloning Template**

| G: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|    | -0.40 | -0.40 | -0.16 | -0.16 | -0.40 | 4 | -0.40 | -0.16 | -0.16 | -0.40 | -0.40 | -0.40 | -0.16 | -0.16 | -0.16 | -0.16 | -0.16 | -0.16 | -0.16 | -0.16 | -0.16 | -0.16 | -0.16 |
| Gₜ: | 0 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 |
|    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Delay Time** \( T = 3 \)

**Example 1: Array Size = 22 x 22**

**Input Image**

**Output Image**
DE-BLURRING CNN
Compensate the blurring effects of an optical device to give a sharply focused output image.

Cloning Template

<table>
<thead>
<tr>
<th>z</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0 0</td>
<td>-0.05 -0.2 -0.3 -0.2 -0.05</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 0 0</td>
<td>-0.2 -0.5 -0.6 -0.5 -0.2</td>
</tr>
<tr>
<td>0</td>
<td>0 10 0 0</td>
<td>-0.3 -0.6 0 -0.6 -0.3</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 0 0</td>
<td>-0.2 -0.5 -0.6 -0.5 -0.2</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 0 0</td>
<td>-0.05 -0.2 -0.3 -0.2 -0.05</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 = 128 x 128

Input Image
Initial State
Output Image
Simulate the well-known illusion where two line segments of identical length but terminated respectively by converging and diverging arrowheads is perceived by humans as if one segment is shorter than the other. For the input image on the left, the CNN must lengthen the upper segment and shorten the lower segment slightly so that the CNN also exhibits the same illusion.

**MULLER-LYER ILLUSION CNN**

**Task**

**Prescription:**

**Cloning Template**

<table>
<thead>
<tr>
<th>-0.1</th>
<th>-0.1</th>
<th>-0.1</th>
<th>-0.1</th>
<th>-0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.1</td>
<td>-0.1</td>
<td>1.3</td>
<td>-0.1</td>
<td>-0.1</td>
</tr>
<tr>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
</tr>
<tr>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
</tr>
</tbody>
</table>

**Boundary Condition**

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

($i*j*$ denotes boundary cells)

**Initial State**

$x_{ij}(0) = u_{ij} = \text{input image}$

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

**Input Image**

**Initial State**

**Output Image**