

FSRCNN 분석

염지현

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

$output\ size = (input\ size - 1) * stride - 2 * padding + dilation * (kernel\ size - 1) + output\ padding + 1$

$output\ size = (3 - 1) * 2 - 2 * 1 + 1 * (3 - 1) + 1 + 1 = 6$

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

1	1.1	1.2
1.3	1.4	1.5
1.6	1.7	1.8

input

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

1	1.1	1.2
1.3	1.4	1.5
1.6	1.7	1.8

input

1	0	1.1	0	1.2
0	0	0	0	0
1.3	0	1.4	0	1.5
0	0	0	0	0
1.6	0	1.7	0	1.8

z = stride - 1
각 행과 열 사이에
z만큼 0을 추가

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

1	1.1	1.2
1.3	1.4	1.5
1.6	1.7	1.8

input

1	0	1.1	0	1.2
0	0	0	0	0
1.3	0	1.4	0	1.5
0	0	0	0	0
1.6	0	1.7	0	1.8

$z = \text{stride} - 1$
각 행과 열 사이에
 z 만큼 0을 추가

0	0	0	0	0	0	0
0	1	0	1.1	0	1.2	0
0	0	0	0	0	0	0
0	1.3	0	1.4	0	1.5	0
0	0	0	0	0	0	0
0	1.6	0	1.7	0	1.8	0
0	0	0	0	0	0	0

$P' = k - p - 1$
 p' 만큼 padding 추가

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

1	1.1	1.2
1.3	1.4	1.5
1.6	1.7	1.8

input

1	0	1.1	0	1.2
0	0	0	0	0
1.3	0	1.4	0	1.5
0	0	0	0	0
1.6	0	1.7	0	1.8

$z = \text{stride} - 1$
각 행과 열 사이에
z만큼 0을 추가

0	0	0	0	0	0	0
0	1	0	1.1	0	1.2	0
0	0	0	0	0	0	0
0	1.3	0	1.4	0	1.5	0
0	0	0	0	0	0	0
0	1.6	0	1.7	0	1.8	0
0	0	0	0	0	0	0

$P' = k - p - 1$
p'만큼 padding 추가

0	0	0	0	0	0	0	0
0	1	0	1.1	0	1.2	0	0
0	0	0	0	0	0	0	0
0	1.3	0	1.4	0	1.5	0	0
0	0	0	0	0	0	0	0
0	1.6	0	1.7	0	1.8	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Output_padding 만큼
아래쪽, 오른쪽에 0으로 패딩 추가

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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

A 6x6 grid of 36 pink squares.

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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

[illegible]

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

[illegible]

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

[illegible]

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Red	Red	Red	Red	Red	White
White	White	White	White	White	White
White	White	White	White	White	White
White	White	White	White	White	White
White	White	White	White	White	White
White	White	White	White	White	White

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

[illegible]

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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

[illegible]

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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

Red	Red	Red	Red	Red	Red
Red	Red	Pink	Pink	Pink	Pink
Pink	Pink	Pink	Pink	Pink	Pink
Pink	Pink	Pink	Pink	Pink	Pink
Pink	Pink	Pink	Pink	Pink	Pink
Pink	Pink	Pink	Pink	Pink	Pink

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Red	Red	Red	Red	Red	Red
Red	Red	Red	Light Pink	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] \rightarrow Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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

Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink
Light Pink	Light Pink	Light Pink	Light Pink	Light Pink	Light Pink

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Red	White
White	White	White	White	White	White
White	White	White	White	White	White
White	White	White	White	White	White
White	White	White	White	White	White

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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

Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Red	Red
Pink	Pink	Pink	Pink	Pink	Pink
Pink	Pink	Pink	Pink	Pink	Pink
Pink	Pink	Pink	Pink	Pink	Pink
Pink	Pink	Pink	Pink	Pink	Pink

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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

A 6x6 grid of squares. The top 5 rows are red, and the bottom row is light pink.

1. ConvTranspose2d 연산 방법

* ConvTranspose2d 연산 예

Input : [1, 1, 3, 3] → Output: [1, 1, 6, 6]

```
m = nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1))
```

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

A 6x6 grid of 36 red squares. Each square is a solid red color with a thin black border. The grid is composed of 6 rows and 6 columns.

2. 실제 nn.ConvTranspose2d 연산 결과 비교

```
* nn.ConvTranspose2d(1, 1, 9, stride=(2,2), padding=(4,4), output_padding=(1,1), bias = False)
```

Jihyun's reuslt

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

*

1	2	3
0	0	0
0	0	0

==

0	0	0
0	0	0
3	2	1

0	0	0	0	0	0
2	4	2	4	2	1
0	0	0	0	0	0
2	4	2	4	2	1
0	0	0	0	0	0
2	4	2	4	2	1

0	0	0	0	0	0
2	4	2	4	2	3
0	0	0	0	0	0
2	4	2	4	2	3
0	0	0	0	0	0
0	0	0	0	0	0

Python result

```
input:
  tensor([[[[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]]]])
torch.Size([1, 1, 3, 3])

Param:
  tensor([[[[1., 2., 3.],
            [0., 0., 0.],
            [0., 0., 0.]]]])
Param shape: torch.Size([1, 1, 3, 3])

output:
  tensor([[[[0., 0., 0., 0., 0., 0.],
            [2., 4., 2., 4., 2., 3.],
            [0., 0., 0., 0., 0., 0.],
            [2., 4., 2., 4., 2., 3.],
            [0., 0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0., 0.]]]], grad_fn=<ThnnConvTranspose2DBackward>)
torch.Size([1, 1, 6, 6])
```

2. 실제 nn.ConvTranspose2d 연산 결과 비교

* nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1), bias = False)

Jihyun's result

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

*

1	1	1
1	1	1
1	1	1

=

1	2	1	2	1	1
2	4	2	4	2	2
1	2	1	2	1	1
2	4	2	4	2	2
1	2	1	2	1	1
1	2	1	2	1	1

Python result

```
input:
  tensor([[[[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]])])
torch.Size([1, 1, 3, 3])

Param:
  tensor([[[[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]])])
Param shape: torch.Size([1, 1, 3, 3])

output:
  tensor([[[[1., 2., 1., 2., 1., 1.],
            [2., 4., 2., 4., 2., 2.],
            [1., 2., 1., 2., 1., 1.],
            [2., 4., 2., 4., 2., 2.],
            [1., 2., 1., 2., 1., 1.],
            [1., 2., 1., 2., 1., 1.]])], grad_fn=<ThnnConvTranspose2DBackward>)
torch.Size([1, 1, 6, 6])
```

2. 실제 nn.ConvTranspose2d 연산 결과 비교

* nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1), bias = False)

Jihyun's result

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	0	0
0	0	0
0	0	0

0	0	0
0	0	0
0	0	1

0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0
0	0	0	0	0	0

Python result

```
input:
  tensor([[[[1., 1., 1.],
            [1., 1., 1.],
            [1., 1., 1.]])])
torch.Size([1, 1, 3, 3])

Param:
  tensor([[[[1., 0., 0.],
            [0., 0., 0.],
            [0., 0., 0.]])])
Param shape: torch.Size([1, 1, 3, 3])

output:
  tensor([[[[0., 0., 0., 0., 0., 0.],
            [0., 1., 0., 1., 0., 0.],
            [0., 0., 0., 0., 0., 0.],
            [0., 1., 0., 1., 0., 0.],
            [0., 0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0., 0.]])], grad_fn=<ThnnConvTranspose2DBackward>)
torch.Size([1, 1, 6, 6])
```


2. 실제 nn.ConvTranspose2d 연산 결과 비교

* nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1), bias = False)

Jihyun's result

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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	1

1	0	0
0	0	0
0	0	0

0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0
0	1	0	1	0	1

Python result

```
input:
tensor([[[[1., 1., 1.],
          [1., 1., 1.],
          [1., 1., 1.]])]])
torch.Size([1, 1, 3, 3])

Param:
tensor([[[[0., 0., 0.],
          [0., 0., 0.],
          [0., 0., 1.]])]])
Param shape: torch.Size([1, 1, 3, 3])

output:
tensor([[[[0., 0., 0., 0., 0., 0.],
          [0., 1., 0., 1., 0., 1.],
          [0., 0., 0., 0., 0., 0.],
          [0., 1., 0., 1., 0., 1.],
          [0., 0., 0., 0., 0., 0.],
          [0., 1., 0., 1., 0., 1.]])]])
grad_fn=<ThnnConvTranspose2DBackward>
torch.Size([1, 1, 6, 6])
```

2. 실제 nn.ConvTranspose2d 연산 결과 비교

* nn.ConvTranspose2d(1, 1, 3, stride=(2,2), padding=(1,1), output_padding=(1,1), bias = False)

Jihyun's result

0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
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
1	0	0
0	0	0

0	0	0
0	0	1
0	0	0

0	1	0	1	0	1
0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0
0	1	0	1	0	1
0	0	0	0	0	0

0	1	0	1	0	0
0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	0	0
0	1	0	1	0	0
0	0	0	0	0	0

Python result

```
input:
tensor([[[[1., 1., 1.],
          [1., 1., 1.],
          [1., 1., 1.]])]])
torch.Size([1, 1, 3, 3])

Param:
tensor([[[[0., 0., 0.],
          [1., 0., 0.],
          [0., 0., 0.]])]])
Param shape: torch.Size([1, 1, 3, 3])

output:
tensor([[[[0., 1., 0., 1., 0., 0.],
          [0., 0., 0., 0., 0., 0.],
          [0., 1., 0., 1., 0., 0.],
          [0., 0., 0., 0., 0., 0.],
          [0., 1., 0., 1., 0., 0.],
          [0., 0., 0., 0., 0., 0.]])]]) grad_fn=<ThnnConvTranspose2DBackward>)
torch.Size([1, 1, 6, 6])
```

2. 실제 nn.ConvTranspose2d 연산 결과 비교

* nn.ConvTranspose2d(1, 1, 9, stride=(2,2), padding=(4,4), output_padding=(1,1), bias = False)
(1920 * 1080) → (3840 * 2160)

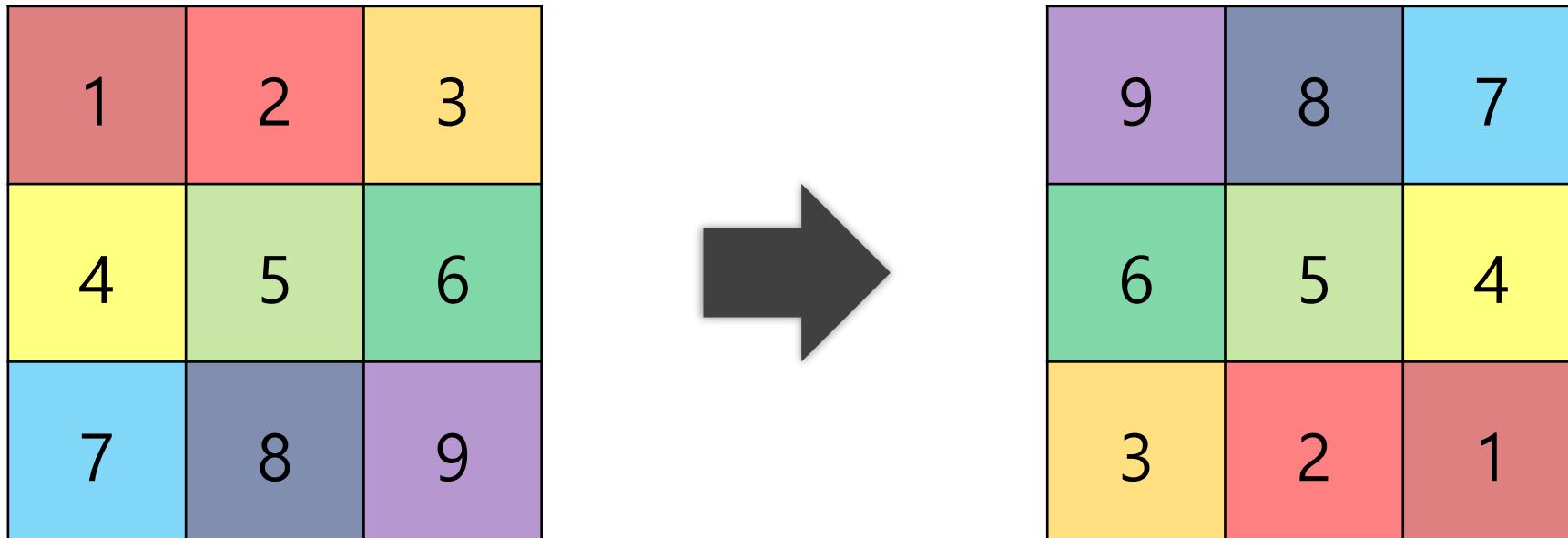
Jihyun's result

```
outuput
[[[ [ 9. 12. 16. ... 18. 21. 14.]
      [ 0.  0.  0. ...  0.  0.  0.]
      [ 9. 12. 16. ... 18. 21. 14.]
      ...
      [ 0.  0.  0. ...  0.  0.  0.]
      [ 0.  0.  0. ...  0.  0.  0.]
      [ 0.  0.  0. ...  0.  0.  0.] ] ] ]
shape: (1, 1, 2160, 3840)
```

Python result

```
Param: tensor([[[[1., 2., 3., 4., 5., 6., 7., 8., 9.],
                  [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., 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.] ] ] ]])
Param shape: torch.Size([1, 1, 9, 9])
input: tensor([[[[1., 1., 1., ..., 1., 1., 1.],
                  [1., 1., 1., ..., 1., 1., 1.],
                  [1., 1., 1., ..., 1., 1., 1.],
                  ...,
                  [1., 1., 1., ..., 1., 1., 1.],
                  [1., 1., 1., ..., 1., 1., 1.],
                  [1., 1., 1., ..., 1., 1., 1.] ] ] ]])
torch.Size([1, 1, 1080, 1920])
tensor([[[[ 9., 12., 16., ..., 18., 21., 14.],
            [ 0.,  0.,  0., ...,  0.,  0.,  0.],
            [ 9., 12., 16., ..., 18., 21., 14.],
            ...,
            [ 0.,  0.,  0., ...,  0.,  0.,  0.],
            [ 0.,  0.,  0., ...,  0.,  0.,  0.],
            [ 0.,  0.,  0., ...,  0.,  0.,  0.] ] ] ]],
        grad_fn=<ThnnConvTranspose2DBackward>)
torch.Size([1, 1, 2160, 3840])
```

3. 결과 분석



결론: Kernel을 상하좌우 바꾼 후 convolution 연산을 진행해야 함

4. 피드백
