

Transposed Convolution

조희철

January 23, 2021

0.1 Transposed Covolution의 이해

♠ Transposed Convolution 1: normal convolution

image의 행과 열 사이에 0으로된 행과 열을 삽입 (dilation) 하고, padding을 더해, intermediate image를 만든 후, 주어진 kernel의 flip과 convolution 연산하는 것으로 transposed convolution을 정의한다.

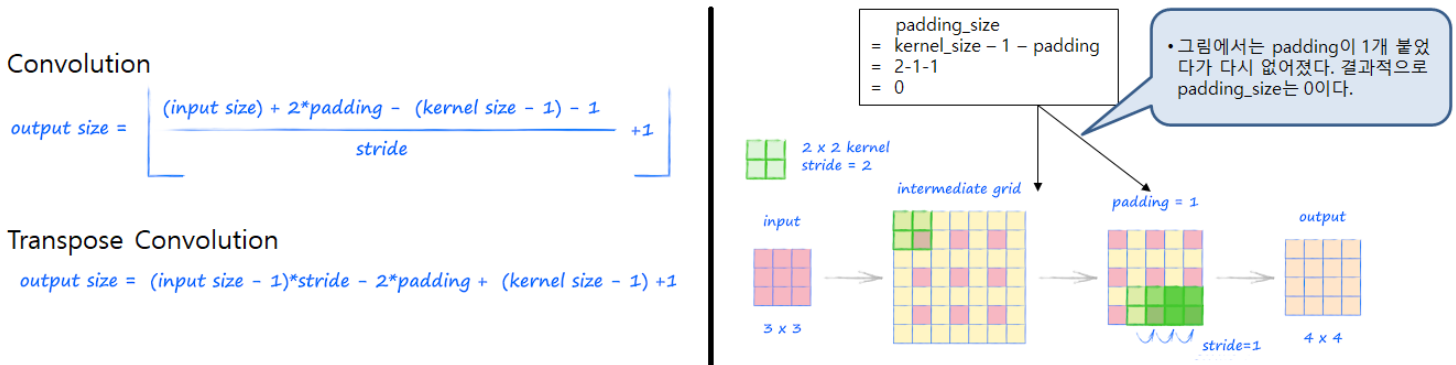


Figure 1: 1. transposed convolution은 입력 이미지를 dilation시키고, padding을 더해 큰 이미지 (intermediate image)로 만든 후, convolution(pad=0, stride=1)을 적용한다.
 2. image의 pixel 사이에 'stride-1'만큼의 0을 삽입한 후, 'kernel-size - 1 - padding' 만큼의 padding를 붙히면, intermediate image가 만들어진다.
 3. transpose convolution에서는 padding 크기가 커질수록 output의 크기는 작아진다. 반면, stride가 커지면 output의 크기도 커진다.
 4. 'kernel-size - 1 - padding'가 0보다 작은 경우에도, output size가 양수이면 된다.
 5. 이렇게 만들어진 intermediate image를 kernel의 flip과 convolution하면 transposed convolution이 된다.

♠ Transposed Convolution 2: kernel과의 행렬곱으로 정의

Transposed convolution이 왜 convolution 연산의 transpose인지 살펴보자. 우선, convolution 연산을 행렬로 변환하는 2가지 방법이 있다.

- im2col: image data의 pixel을 중복하고, weight는 reshape만 한다. 이 방법은 메모리를 많이 사용하지만, 계산적인 측면에서 효율적이다.
- image flatten: $H \times W$ image는 reshape 하여 $1 \times (H \times W)$ 로 변환하고 weight는 $(H \times W) \times (OH \times OW)$ 로 변환한다. 변환된 2개의 행렬을 곱한 결과가 convolution 연산이 된다.

여기서는 image flatten 방식으로 convolution을 정의하고, 이로 부터 transposed convolution을 정의해 보자. 예를 들어서 살펴보자.

$$\text{image } X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{pmatrix} \quad \text{kernel } W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$$

image X 를 flatten하여

$$X_f := (x_{11} \ x_{12} \ x_{13} \ x_{14} \ x_{21} \ x_{22} \ x_{23} \ x_{24} \ x_{31} \ x_{32} \ x_{33} \ x_{34} \ x_{41} \ x_{42} \ x_{43} \ x_{44}),$$

W 를 재배치하여 W_r (아래 첨자 r 을 붙혔다)을 그림 (2)와 같이 정의하자.

X_f, W_r 을 이렇게 정의하고 (행렬) 곱을 한 후, reshape하면 convolution 연산 결과가 된다. 즉,

$$\text{Conv}(X, W) := X_f W_r \rightarrow \text{reshape } (OH \times OW) = (3 \times 3)$$

$$W_r = \begin{bmatrix} \boxed{\begin{matrix} w_{11} & 0 & 0 \\ w_{12} & w_{11} & 0 \\ 0 & w_{12} & w_{11} \\ 0 & 0 & w_{12} \end{matrix}} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \\ \boxed{\begin{matrix} w_{21} & 0 & 0 \\ w_{22} & w_{21} & 0 \\ 0 & w_{22} & w_{21} \\ 0 & 0 & w_{22} \end{matrix}} & \boxed{\begin{matrix} w_{11} & 0 & 0 \\ w_{12} & w_{11} & 0 \\ 0 & w_{12} & w_{11} \\ 0 & 0 & w_{12} \end{matrix}} & \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & \boxed{\begin{matrix} w_{21} & 0 & 0 \\ w_{22} & w_{21} & 0 \\ 0 & w_{22} & w_{21} \\ 0 & 0 & w_{22} \end{matrix}} & \boxed{\begin{matrix} w_{11} & 0 & 0 \\ w_{12} & w_{11} & 0 \\ 0 & w_{12} & w_{11} \\ 0 & 0 & w_{12} \end{matrix}} \\ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} & \boxed{\begin{matrix} w_{21} & 0 & 0 \\ w_{22} & w_{21} & 0 \\ 0 & w_{22} & w_{21} \\ 0 & 0 & w_{22} \end{matrix}} & \boxed{\begin{matrix} w_{11} & 0 & 0 \\ w_{12} & w_{11} & 0 \\ 0 & w_{12} & w_{11} \\ 0 & 0 & w_{12} \end{matrix}} \\ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} & \boxed{\begin{matrix} w_{21} & 0 & 0 \\ w_{22} & w_{21} & 0 \\ 0 & w_{22} & w_{21} \\ 0 & 0 & w_{22} \end{matrix}} \end{bmatrix}$$

Figure 2: convolution kernel reshape. $((H \times W) \times (OH \times OW)) = (16 \times 9)$

이제, 그림 (1)에서와 다른 방식의 transposed convolution을 정의해 보자. $(OH \times OW) = (3 \times 3)$ 크기의 임의의 image Y 와 kernel W 가 주어졌을 때,

$$\text{TransposedConv}(Y, W) := Y_f W_r^T$$

를 Y, W 의 transposed convolution으로 정의하자. 이렇게 transposed convolution을 정의하면, 그림 (1)에서의 정의와 동일한 결과가 된다. 그 이유를 알아보자. 먼저 그림 (3)에서 W_r^T 를 관측해 보자. 이 행렬은 어떤 2×2 행렬의 (그림 (2)에서 정의한) 재배치 행렬 (\blacksquare_r)의 일부이다. 정답은 바로 $W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$ 의 flip 행렬이다.

$$\text{flip}(W) := \begin{pmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{pmatrix}$$

$$W_r^T = \begin{bmatrix} \boxed{\begin{matrix} w_{11} & w_{12} & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 \\ 0 & 0 & w_{11} & w_{12} \end{matrix}} & \boxed{\begin{matrix} w_{21} & w_{22} & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & w_{21} & w_{22} \end{matrix}} & \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} & \boxed{\begin{matrix} w_{11} & w_{12} & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 \\ 0 & 0 & w_{11} & w_{12} \end{matrix}} & \boxed{\begin{matrix} w_{21} & w_{22} & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & w_{21} & w_{22} \end{matrix}} & \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} & \boxed{\begin{matrix} w_{11} & w_{12} & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 \\ 0 & 0 & w_{11} & w_{12} \end{matrix}} & \boxed{\begin{matrix} w_{21} & w_{22} & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & w_{21} & w_{22} \end{matrix}} \end{bmatrix}$$

Figure 3: W_r 을 transpose한 matrix

좀 더 살펴보기 위해, W_r 의 transpose matrix W_r^T 를 변형한 $\overline{W_r^T}$ 를 그림 (4)과 같이 정의하자. 그림 (4)에서 정의한 $\overline{W_r^T}$ 는 $\text{flip}(W)$ 의 재배치(그림 (2)에서 정의) 행렬 $\text{flip}(W)_r$ 과 일치함을 알 수 있다. 즉,

$$\text{flip}(W)_r = \overline{W_r^T}$$

이제 정리를 해보자.

- transposed convolution을 $W_r^T (9 \times 16)$ 를 이용해서 정의했고, 이 W_r^T 를 확장하여 25×16 크기의 행렬 $\overline{W_r^T}$ 를 만들었다. $\overline{W_r^T}$ 를 만들면서 추가된 부분이 계산에 영향을 미치지 않고, 이미지 Y_f 와 행렬곱이 될 수 있도록 해야한다. 그래서 Y_f 에 padding을 붙혀 $\overline{Y_f}$ 를 만든다.

$$\overline{W_r^T} =$$

w_{22}	0	0	0				
w_{21}	w_{22}	0	0				
0	w_{21}	w_{22}	0				
0	0	w_{21}	w_{22}				
0	0	0	w_{21}				
w_{12}	0	0	0	w_{22}	0	0	0
w_{11}	w_{12}	0	0	w_{21}	w_{22}	0	0
0	w_{11}	w_{12}	0	0	w_{21}	w_{22}	0
0	0	w_{11}	w_{12}	0	0	w_{21}	w_{22}
0	0	0	w_{11}	0	0	0	w_{21}
				w_{12}	0	0	0
				w_{11}	w_{12}	0	0
				0	w_{11}	w_{12}	0
				0	0	w_{11}	w_{12}
				0	0	0	w_{11}
				w_{12}	0	0	0
				w_{11}	w_{12}	0	0
				0	w_{11}	w_{12}	0
				0	0	w_{11}	w_{12}
				0	0	0	w_{11}
				w_{12}	0	0	0
				w_{11}	w_{12}	0	0
				0	w_{11}	w_{12}	0
				0	0	w_{11}	w_{12}
				0	0	0	w_{11}
				w_{12}	0	0	0
				w_{11}	w_{12}	0	0
				0	w_{11}	w_{12}	0
				0	0	w_{11}	w_{12}
				0	0	0	w_{11}
				w_{12}	0	0	0
				w_{11}	w_{12}	0	0
				0	w_{11}	w_{12}	0
				0	0	w_{11}	w_{12}
				0	0	0	w_{11}

Figure 4: W_r^T 를 변형한 행렬. 이 $\overline{W_r^T}$ 행렬을 관찰해 보면, $\text{flip}(W)_r$ 임을 알 수 있다.

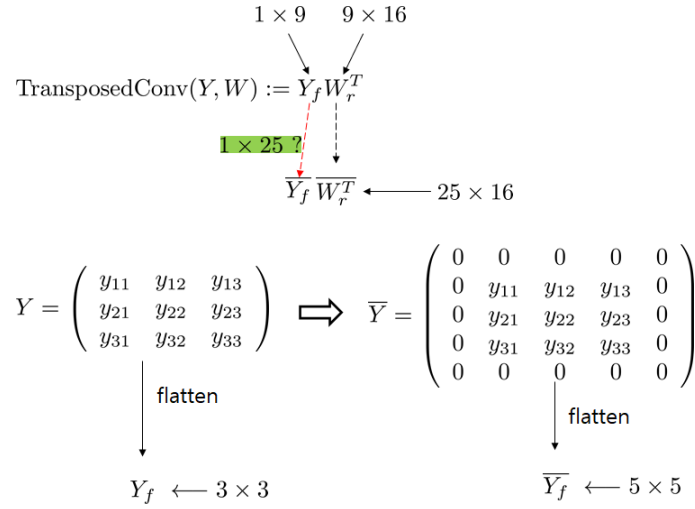


Figure 5: transposed convolution 전개 과정에서 행렬 크기의 변화.

$$\begin{aligned}
\text{TransposedConv}(Y, W) &= Y_f W_r^T \leftarrow (1 \times 9) (9 \times 13) \\
&= \overline{Y_f} \overline{W_r^T} \leftarrow (1 \times 25) (25 \times 13) \\
&= \overline{Y_f} \text{flip}(W)_r \\
&= \text{Conv}(\overline{Y_f}, \text{flip}(W))
\end{aligned}$$

♠ Transposed Convolution 3: kernel 전개

cs231n 강의¹에서 transposed convolution을 kernel들에 대한 weighted sum으로 설명하고 있다. 이 때, weight는 각 pixel 값들이 된다.

이렇게 transposed convolution을 계산하면, 앞에서 정의한 결과와 동일하게 되는데, 그 이유는 그림 (4)를 관측해 보면, 알 수 있다. $\overline{W_r^T}$ 의 모든 행은 $(w_{11}, w_{12}, 0, 0, w_{21}, w_{22})$ 를 포함하고 있다. $\overline{Y_f}$ 와 곱하는 과정에서 각각의 행은 Y 의 y_{ij} 와 순차적으로 곱해진다.

¹http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture12.pdf

- image의 각 pixel값은 kernel과 곱해진다.
- transposed convolution은 kernel을 이어 붙히는 방식이다.

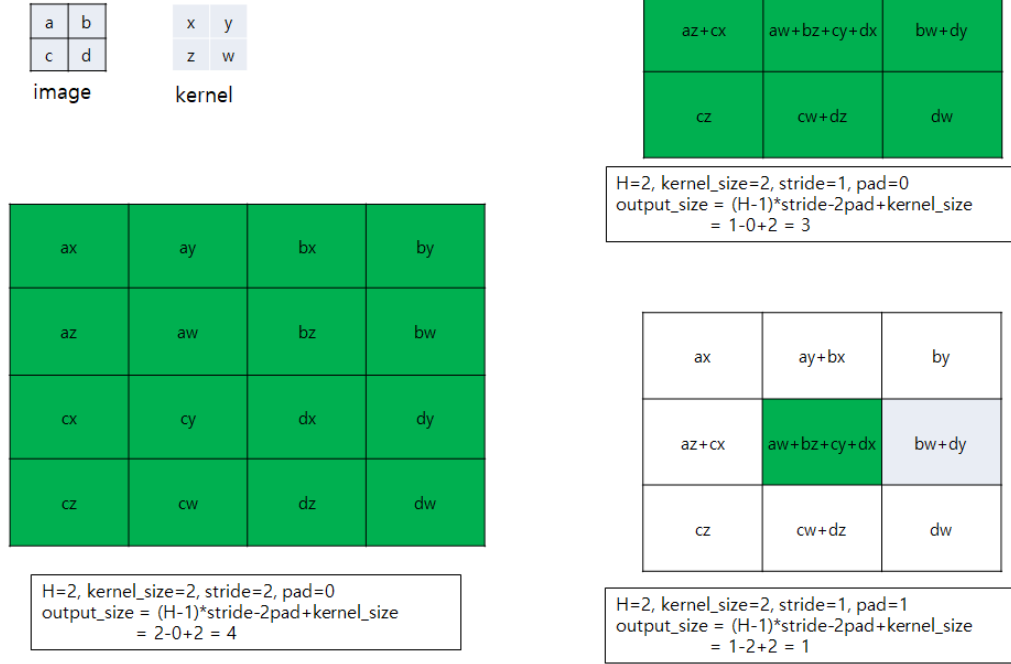
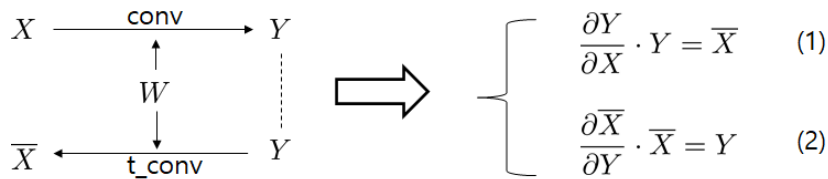


Figure 6: kernel 전개: ouptput 이미지의 크기를 (stride, padding을 고려하여) 미리 계산해 놓고, 각 cell의 값을 kernel과 하나의 pixel곱으로 더해나가는 방식. 모든 pixel에 대하여 반복적으로 수행하면 된다.

♠ Transposed Convolution 4: backward pass

Transposed Convolution을 Convolution의 입력 이미지에 대한 미분으로 보는 관점. 이런 관점은 Transposed Convolution의 정의가 아니라, 성질이라고 봐야한다.



- Figure 7: 1. transposed convolution을 convolution의 backward로 이해할 수도 있다. 그림의 식(1)에서 Y, 식(2)에서 \bar{X} 는 넘어온 gradient로 이해하면 된다.
2. 임의의 Y, W가 주어져 있을 때, transposed convolution \bar{X} 를 식(1)로 구할 수는 없다. X를 알 수 없기 때문이다.
 3. 식(1)은 행렬곱의 backpropagation을 생각해 보면 알 수 있다.
 4. 식(1),(2)는 X, W를 2×2 행렬에 대해서 계산해 보면, 이해하는데 도움이 된다.

$$X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} \quad W = \begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix}$$

$$\text{Conv}(X, W) = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4 =: Y$$

$$\text{TransposedConv}(Y, W) = Y \begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix} = Y \frac{\partial Y}{\partial X}$$

$\text{flip}(W) = \begin{pmatrix} w_4 & w_3 \\ w_2 & w_1 \end{pmatrix}$
 \otimes
 $\begin{pmatrix} 0 & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Figure 8: 2×2 행렬에서 convolution의 backward가 transposed convolution임을 확인할 수 있다.