Deep Learning

7.1 Transposed Convolutions

Dr. Konda Reddy Mopuri
kmopuri@iittp.ac.in
Dept. of CSE, IIT Tirupati

Transposed Convolution

1. Deep generative architectures need layers that increase the signal dimensions

Transposed Convolution

1. Deep generative architectures need layers that increase the signal dimensions
2. This is contrary to what we have seen

Transposed Convolution

1. Deep generative architectures need layers that increase the signal dimensions
2. This is contrary to what we have seen
3. Convolutions, pooling, etc. reduce the signal dimension

Transposed Convolution

1. Deep generative architectures need layers that increase the signal dimensions
2. This is contrary to what we have seen
3. Convolutions, pooling, etc. reduce the signal dimension
4. Transposed Convolutions increase the output size

Revisit Convolution in deep learning

1. 1D convolution $x \circledast y$

$$
\begin{aligned}
y_i &= \sum_a x_{i+a-1} k_a \\
&= \sum_u x_u k_{u-i+1}
\end{aligned}
$$

Revisit Convolution in deep learning

1. 1D convolution $x \circledast y$

$$y_i = \sum_a x_{i+a-1} k_a$$
$$= \sum_u x_u k_{u-i+1}$$

2. Backpropagating through convolution

$$\left[\frac{\partial l}{\partial x}\right]_u = \left[\frac{\partial l}{\partial x_u}\right]$$
$$= \sum_i \frac{\partial l}{\partial y_i} \frac{\partial y_i}{\partial x_u}$$
$$= \sum_i \frac{\partial l}{\partial y_i} k_{u-i+1}$$

Revisit Convolution in deep learning

1. 1D convolution $x \circledast y$

$$\begin{aligned} y_i &= \sum_a x_{i+a-1} k_a \\ &= \sum_u x_u k_{u-i+1} \end{aligned}$$

2. Backpropagating through convolution

$$\begin{aligned} \left[\frac{\partial l}{\partial x}\right]_u &= \left[\frac{\partial l}{\partial x_u}\right] \\ &= \sum_i \frac{\partial l}{\partial y_i} \frac{\partial y_i}{\partial x_u} \\ &= \sum_i \frac{\partial l}{\partial y_i} k_{u-i+1} \end{aligned}$$

3. This also looks like convolution, but the kernel is visited in the reverse order

Backpropagating through Convolution

1. Backpropagating through convolution

$$\left[\frac{\partial l}{\partial x}\right]_u = \left[\frac{\partial l}{\partial x_u}\right] = \sum_i \frac{\partial l}{\partial y_i}\frac{\partial y_i}{\partial x_u} = \sum_i \frac{\partial l}{\partial y_i} k_{u-i+1}$$

2. This is the standard convolution operation from the signal processing

$$(x * k)_i = \sum_a x_a k_{i-a+1}$$

Backpropagating through Convolution

1. Backpropagating through convolution

$$\left[\frac{\partial l}{\partial x}\right]_u = \left[\frac{\partial l}{\partial x_u}\right] = \sum_i \frac{\partial l}{\partial y_i}\frac{\partial y_i}{\partial x_u} = \sum_i \frac{\partial l}{\partial y_i} k_{u-i+1}$$

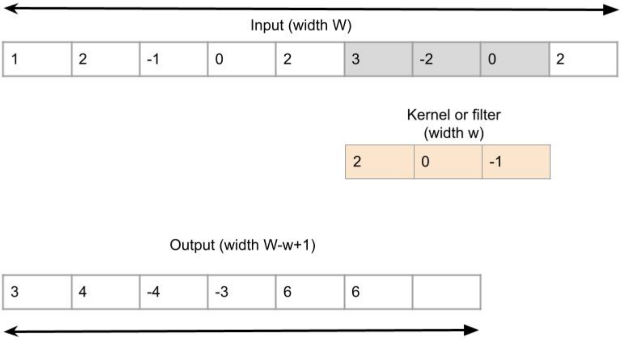
2. This is the standard convolution operation from the signal processing

$$(x * k)_i = \sum_a x_a k_{i-a+1}$$

3. Therefore the backprop in convolution becomes

$$\text{if } y = x \circledast k,$$
$$\text{then } \left[\frac{\partial l}{\partial x}\right] = \left[\frac{\partial l}{\partial y}\right] * k$$

1D convolution example



1D Convolution as matrix operation

$$\begin{bmatrix} 2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -1 \\ 0 \\ 2 \\ 3 \\ -2 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ -4 \\ -3 \\ 6 \\ 6 \\ -6 \end{bmatrix}$$

Backprop as a matrix operation

- Backpropagation (i.e, the signal processing convolution) can also be implemented as a matrix operation.

Backprop as a matrix operation

- Backpropagation (i.e, the signal processing convolution) can also be implemented as a matrix operation.
- Reversing the kernel can be achieved via transposing the weight matrix.

Backprop as a matrix operation

- Backpropagation (i.e, the signal processing convolution) can also be implemented as a matrix operation.
- Reversing the kernel can be achieved via transposing the weight matrix.
- $$\begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix}^T = \begin{bmatrix} k_1 & 0 & 0 & 0 & 0 \\ k_2 & k_1 & 0 & 0 & 0 \\ k_3 & k_2 & k_1 & 0 & 0 \\ 0 & k_3 & k_2 & k_1 & 0 \\ 0 & 0 & k_3 & k_2 & k_1 \\ 0 & 0 & 0 & k_3 & k_2 \\ 0 & 0 & 0 & 0 & k_3 \end{bmatrix}$$

Backprop as a matrix operation

- Backpropagation (i.e, the signal processing convolution) can also be implemented as a matrix operation.
- Reversing the kernel can be achieved via transposing the weight matrix.
- $$\begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix}^T = \begin{bmatrix} k_1 & 0 & 0 & 0 & 0 \\ k_2 & k_1 & 0 & 0 & 0 \\ k_3 & k_2 & k_1 & 0 & 0 \\ 0 & k_3 & k_2 & k_1 & 0 \\ 0 & 0 & k_3 & k_2 & k_1 \\ 0 & 0 & 0 & k_3 & k_2 \\ 0 & 0 & 0 & 0 & k_3 \end{bmatrix}$$
- Therefore, this operation is referred to as Transposed Convolution.

Backprop as a matrix operation

- Backpropagation (i.e, the signal processing convolution) can also be implemented as a matrix operation.
- Reversing the kernel can be achieved via transposing the weight matrix.
- $$\begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 & 0 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix}^T = \begin{bmatrix} k_1 & 0 & 0 & 0 & 0 \\ k_2 & k_1 & 0 & 0 & 0 \\ k_3 & k_2 & k_1 & 0 & 0 \\ 0 & k_3 & k_2 & k_1 & 0 \\ 0 & 0 & k_3 & k_2 & k_1 \\ 0 & 0 & 0 & k_3 & k_2 \\ 0 & 0 & 0 & 0 & k_3 \end{bmatrix}$$
- Therefore, this operation is referred to as Transposed Convolution.
- F.conv_transpose1d implements this operation.

Transposed Convolution

1. Increases the signal dimension

Transposed Convolution

1. Increases the signal dimension
2. Used in Deep generative models