# Back-Propagation: From Fully Connected to Convolutional Layers

**Shuiwang Ji**
Texas A&M University
College Station, TX 77843
sji@tamu.edu

**Yaochen Xie**
Texas A&M University
College Station, TX 77843
ethanycx@tamu.edu

## 1 Introduction

This note aims at providing an introduction to the back-propagation (BP) for the convolution. We derive the back-propagation for the convolution from the general case and further show that the convolutional layer is a particular case of fully-connected layer. This document is based on lecture notes by Shuiwang Ji at Texas A&M University and can be used for undergraduate and graduate level classes.

## 2 Back-Propagation in Fully Connected Layers

### 2.1 Forward-Propagation and Back-Propagation in General

In a general $\mathcal{L}$-layer stacked neural network, we let $\boldsymbol{\theta}_k \in \mathbb{R}^{\tilde{d}_k}$ denote the parameters (formatted as column vectors) at layer $k$, and $\boldsymbol{x}^{(k)} \in \mathbb{R}^{d_k}$ denote the outputs (formatted as column vectors) of layer $k$. The relationship between the input $\boldsymbol{x} =: \boldsymbol{x}^{(0)}$ and the final output (hypothesis) $\boldsymbol{h(x)} = \boldsymbol{x}^{(\mathcal{L})}$ can be represented by a composed function consisting of a series of functions

$$\boldsymbol{f}_{\boldsymbol{\theta}_1} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1},\ \boldsymbol{f}_{\boldsymbol{\theta}_2} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2},\ \cdots,\ \boldsymbol{f}_{\boldsymbol{\theta}_\mathcal{L}} : \mathbb{R}^{d_{\mathcal{L}-1}} \to \mathbb{R}^{d_\mathcal{L}},$$

as

$$\boldsymbol{h}(\boldsymbol{x}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_\mathcal{L}) = \boldsymbol{f}_{\boldsymbol{\theta}_\mathcal{L}}(\cdots(\boldsymbol{f}_{\boldsymbol{\theta}_2}(\boldsymbol{f}_{\boldsymbol{\theta}_1}(\boldsymbol{x}))) =: \boldsymbol{f}_{\boldsymbol{\theta}_\mathcal{L}} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{\theta}_2} \circ \boldsymbol{f}_{\boldsymbol{\theta}_1}(\boldsymbol{x}), \tag{1}$$

where the parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots \boldsymbol{\theta}_\mathcal{L}$ are within the functions.

The in-sample error $e$ is yet another function of $\boldsymbol{h}(\boldsymbol{x}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots)$. While applying the gradient descent to the $e$, we are taking the partial derivative of $e$ with respect to the parameters $\boldsymbol{\theta}$. Naturally, the chain rule turns out to be useful when calculating the derivative, i.e.,

$$\begin{aligned}
\frac{\partial e}{\partial \boldsymbol{\theta}_k} &= \frac{\partial e}{\partial \boldsymbol{x}^{(\mathcal{L})}} \frac{\partial \boldsymbol{x}^{(\mathcal{L})}}{\partial \boldsymbol{\theta}_k} \\
&= \frac{\partial e}{\partial \boldsymbol{x}^{(\mathcal{L})}} \frac{\partial \boldsymbol{x}^{(\mathcal{L})}}{\partial \boldsymbol{x}^{(k)}} \frac{\partial \boldsymbol{x}^{(k)}}{\partial \boldsymbol{\theta}_k} && (2) \\
&= \frac{\partial e}{\partial \boldsymbol{x}^{(\mathcal{L})}} \frac{\partial \boldsymbol{x}^{(\mathcal{L})}}{\partial \boldsymbol{x}^{(\mathcal{L}-1)}} \frac{\partial \boldsymbol{x}^{(\mathcal{L}-1)}}{\partial \boldsymbol{x}^{(\mathcal{L}-2)}} \cdots \frac{\partial \boldsymbol{x}^{(k+1)}}{\partial \boldsymbol{x}^{(k)}} \frac{\partial \boldsymbol{x}^{(k)}}{\partial \boldsymbol{\theta}_k},\ k = 1 \cdots \mathcal{L} && (3)
\end{aligned}$$

where $\boldsymbol{x}^{(k)} = \boldsymbol{f}_{\boldsymbol{\theta}_k}(\cdots(\boldsymbol{f}_{\boldsymbol{\theta}_2}(\boldsymbol{f}_{\boldsymbol{\theta}_1}(\boldsymbol{x})))$ is the output of the k-th layer.

### 2.2 Forward-Propagation and Back-Propagation in Fully Connected Layers

We follow the notations in [1] below. Namely, we let $\boldsymbol{W}_k \in \mathbb{R}^{(d_{k-1}+1) \times d_k}$ denote the weight matrix, where the bias terms are contained in an additional dimension of layer $k-1$, let $\boldsymbol{s}^{(k)}$ denote

the incoming signal of layer $k$, and let $\theta$ denote the activation function. In a fully-connected layer in particular, the output of layer $k$ w.r.t. the output of layer $k-1$ given by function $\boldsymbol{f}_{\boldsymbol{W}_k} : \mathbb{R}^{d_{k-1}+1} \rightarrow \mathbb{R}^{d_k+1}$ is

$$\boldsymbol{x}^{(k)} = \boldsymbol{f}_{\boldsymbol{W}_k}(\boldsymbol{x}^{(k-1)}) = \begin{bmatrix} 1 \\ \theta(\boldsymbol{s}^{(k)}) \end{bmatrix} \tag{4}$$

$$\boldsymbol{s}^{(k)} = (\boldsymbol{W}_k)^T \boldsymbol{x}^{(k-1)} \tag{5}$$

Following the chain rule described in Eq. (2) , the partial derivative of $e$ with respect to $\boldsymbol{W}_k$ in the fully-connected case can be derived from

$$\frac{\partial e}{\partial w_{ij}^{(k)}} = \frac{\partial s_j^{(k)}}{\partial w_{ij}^{(k)}} \frac{\partial e}{\partial s_j^{(k)}} := x_i^{(k-1)} \delta_j^{(k)}, \tag{6}$$

with the fact that for a single weight $w_{ij}^{(k)}$, a change in $w_{ij}^{(k)}$ only affects $\boldsymbol{s}_j^{(k)}$. Therefore we have

$$\frac{\partial e}{\partial \boldsymbol{W}_k} = \boldsymbol{x}^{(k-1)} (\boldsymbol{\delta}^{(k)})^T, \tag{7}$$

where $\boldsymbol{\delta}^{(k)} = \frac{\partial e}{\partial \boldsymbol{s}^{(k)}}$ is called the sensitivity vector for layer $k$. Based on the chain rule, the sensitivity vector can be further computed as

$$\boldsymbol{\delta}^{(k)} = \theta'(\boldsymbol{s}^{(k)}) \otimes [\boldsymbol{W}^{(k+1)} \boldsymbol{\delta}^{(k+1)}]_1^{d^{(k)}} \tag{8}$$

where $\otimes$ denotes the element-wise multiplication and $[\boldsymbol{W}^{(k+1)} \boldsymbol{\delta}^{(k+1)}]_1^{d^{(k)}}$ excludes the first element of $\boldsymbol{W}^{(k+1)} \boldsymbol{\delta}^{(k+1)}$. Then we are able to recursively compute the derivative from $\mathcal{L}$ to 1.

The weights $\boldsymbol{W}_k$ are used in both the forward-propagation when we compute the output of each layer from the output of previous layer, as shown in Eq. (5) , and the back-propagation, when we compute the sensitivity vector of current layer from the sensitivity vector of the next layer, as shown in Eq. (8). As you may have noticed, the weight matrix is **transposed** in the forward-propagation Eq. (5) but **not transposed** in the back-propagation Eq. (8). We will find it similar but different in the convolution case.

## 3 Back-Propagation in Convolutional Layers

In this section, we will first introduce the forward-propagation and back-propagation of the convolution in the single channel case. Then we will discuss how the convolution can be a particular case of the fully connected. At the end of this section, we will further discuss how the convolution works with multiple input and output channels.

In this section, we refer to the outputs $\boldsymbol{x}^{(k)}$ at current layer as the outputs of the layer $k$, and the outputs $\boldsymbol{x}^{(k-1)}$ at the previous layer as the input of layer $k$. And for simplicity, we ignore the bias terms, i.e., $\boldsymbol{w}_{ij}^{(k)} \in \mathbb{R}^{d_k}$ instead of $\mathbb{R}^{d_k+1}$ where $d_k$ denotes the number of channels of the inputs to layer $k$, which makes Eq. (9) different from Eq. (4).

### 3.1 Forward-Propagation with Single Channel

Let $s_{pq}^{(k)}$ denote the location $(p, q)$ of the incoming signal $\boldsymbol{s}$ at layer $k$ and $x_{pq}^{(k)}$ denote the location $(p, q)$ of outputs $\boldsymbol{x}$ at layer $k$. Then in a forward propagation we have:

$$x_{pq}^{(k)} = \theta(s_{pq}^{(k)}), \tag{9}$$

$$s_{pq}^{(k+1)} = \sum_{i\in\mathcal{K}}\sum_{j\in\mathcal{K}} x_{p+i,q+j}^{(k)} w_{ij} =: \sum_{i\in\mathcal{K},j\in\mathcal{K}} x_{p+i,q+j}^{(k)} w_{ij}, \tag{10}$$

for all $(p, q)$, where $w_{ij}$ denotes weight at location $(i, j)$ in filter $F$, and $\mathcal{K}$ denotes a set of neighbour. For instances, in a $3 \times 3$ convolution, $\mathcal{K} = \{-1, 0, 1\}$; and in a $5 \times 5$ convolution, $\mathcal{K} = \{-2, -1, 0, 1, 2\}$.

### 3.2 Back-Propagation with Single Channel

During the back-propagation in a convolutional layer, we have

$$\frac{\partial e}{\partial x_{pq}^{(k)}} = \sum_{i\in\mathcal{K},j\in\mathcal{K}} \frac{\partial e}{\partial s_{p-i,q-j}^{(k+1)}} \cdot \frac{\partial s_{p-i,q-j}^{(k+1)}}{\partial x_{pq}^{(k)}} \tag{11}$$

$$= \sum_{i\in\mathcal{K},j\in\mathcal{K}} \frac{\partial e}{\partial s_{p-i,q-j}^{(k+1)}} \cdot \frac{\partial s_{p,q}^{(k+1)}}{\partial x_{p+i,q+j}^{(k)}} \tag{12}$$

$$= \sum_{i\in\mathcal{K},j\in\mathcal{K}} \frac{\partial e}{\partial s_{p-i,q-j}^{(k+1)}} \cdot w_{ij} \tag{13}$$

$$= \sum_{i\in\mathcal{K},j\in\mathcal{K}} \frac{\partial e}{\partial s_{p+i,q+j}^{(k+1)}} \cdot w_{-i,-j} \tag{14}$$

Therefore, the sensitivity vector is computed as

$$\delta_{pq}^{(k)} = \frac{\partial e}{\partial s_{pq}^{(k)}} = \theta'(s_{pq}^{(k)}) \cdot \frac{\partial e}{\partial x_{pq}^{(k)}} \tag{15}$$

$$= \theta'(s_{pq}^{(k)}) \cdot \left( \sum_{i\in\mathcal{K},j\in\mathcal{K}} \frac{\partial e}{\partial s_{p+i,q+j}^{(k+1)}} \cdot w_{-i,-j} \right) \tag{16}$$

$$= \theta'(s_{pq}^{(k)}) \cdot \left( \sum_{i\in\mathcal{K},j\in\mathcal{K}} \delta_{p+i,q+j}^{(k+1)} \cdot w_{-i,-j} \right) \tag{17}$$

We can see that the indexes of $w$ are opposite in FP and BP, which shows that BP uses the **inverted** (**not transposed**) filter in FP. This differs from what we observed and concluded at the end of section 2.2. In the following subsection, we will discuss why this happens. This was also discussed in [2].

### 3.3 Connection between Convolution and the Fully Connected

Long story short, the convolutional layer is a particular case of the fully connected layer whose weights are shared and non-zero only at its local neighbors. We illustrate this in a 1D-convolution (with padding) case in figure 1a.

From a fully connected point of view, the weight matrix $\tilde{W}$ of the convolution can be obtained by sliding the kernel from one side to another. And the weights are zero outside the kernel. In this case, we re-write the forward-propagation as follow

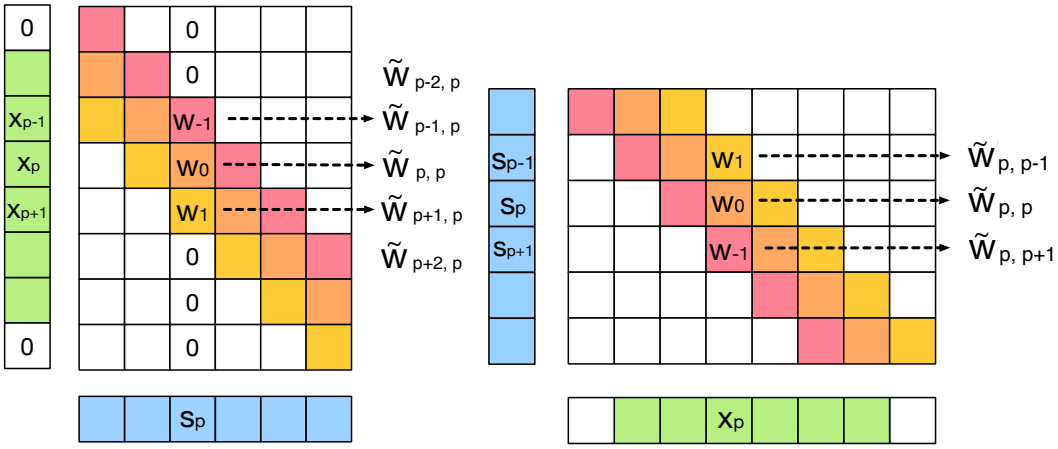
$$s_p = \sum_{i\in\mathcal{K}} x_{p+i} w_i \tag{18}$$

$$= x_{p-1}w_{-1} + x_p w_0 + x_{p+1} w_1 \tag{19}$$

$$= x_{p-1}\tilde{w}_{p,p-1} + x_p \tilde{w}_{p,p} + x_{p+1}\tilde{w}_{p,p+1} \tag{20}$$

$$= \cdots + x_{p-2} \cdot 0 + x_{p-1}\tilde{w}_{p,p-1} + x_p \tilde{w}_{p,p} + x_{p+1}\tilde{w}_{p,p+1} + x_{p+2} \cdot 0 + \cdots \tag{21}$$

$$=: (\tilde{\boldsymbol{W}}^T)_p . \mathbf{x} \tag{22}$$




**(a)** Forward-Propagation **(b)** Back-Propagation

**Figure 1: Convolution as a fully connected**: here $w_i$ denotes the weight at location $i$ in a kernel and $\tilde{w}_{i,j}$ denotes the weight at location $(i, j)$ in the weight matrix $\tilde{W}$ in terms of the fully connected perspective. The green vector denotes the (padded) input $\boldsymbol{x}^{(k-1)}$ and the blue vector denotes the incoming signal $\boldsymbol{s}^{(k-1)}$ at layer $k$. **Note that to better show the correspondence between the locations, the matrix shown in both figure are put transposed to which we use in Eq. 23 and Eq. 28**.

where $w_i$ denotes the weights in the kernel, $\tilde{w}_{j,k}$ denotes the weights in the fully-connected weight matrix $\tilde{\boldsymbol{W}}^T$, and $(\tilde{\boldsymbol{W}}^T)_{p\cdot}$ denotes the $p$-th row of the transposed weight matrix $\tilde{\boldsymbol{W}}^T$, i.e. the $p$-th column of the matrix $\tilde{\boldsymbol{W}}$. Thus we have

$$\boldsymbol{s} = \tilde{\boldsymbol{W}}^T \mathbf{x} \tag{23}$$

Similarly, we can re-write the back-propagation process as

$$\begin{aligned}
\frac{\partial e}{\partial x_p} &= \sum_{i \in \mathcal{K}} \frac{\partial e}{\partial s_{p+i}} \cdot w_{-i} && (24)\\
&= \frac{\partial e}{\partial s_{p+1}} \cdot w_{-1} + \frac{\partial e}{\partial s_p} \cdot w_0 + \frac{\partial e}{\partial s_{p-1}} \cdot w_1 && (25)\\
&= \frac{\partial e}{\partial s_{p+1}} \cdot \tilde{w}_{p+1,p} + \frac{\partial e}{\partial s_p} \cdot \tilde{w}_{p,p} + \frac{\partial e}{\partial s_{p-1}} \cdot \tilde{w}_{p-1,p} && (26)\\
&=: \tilde{\boldsymbol{W}}_{p\cdot} \boldsymbol{\delta} && (27)
\end{aligned}$$

where $\tilde{\boldsymbol{W}}_{p\cdot}$ denotes the $p$-th row of the weight matrix $\tilde{\boldsymbol{W}}$, which implies

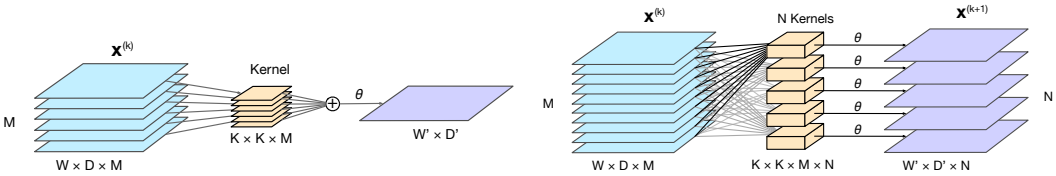
$$\boldsymbol{\delta}^{(k)} = \theta'(\boldsymbol{s}^{(k)}) \otimes (\tilde{\boldsymbol{W}}^{(k+1)} \boldsymbol{\delta}^{(k+1)}) \tag{28}$$

Now we can see that the Eq. (23) is consistent to Eq. (5) and Eq. (28) is a simplified version (without bias) of Eq. (8).

### 3.4 Multiple Input and Output Feature Maps

The case of multiple input and output feature maps is described in [2]. In particular, for M input channels and N output channels, the forward-propagation becomes

$$\boldsymbol{x}_{pq}^{(k)} = \theta(\boldsymbol{s}_{pq}^{(k)}), \tag{29}$$




**(a)** Multi-channel input and single-channel output

**(b)** Multi-channel input and output

**Figure 2: Convolution with multiple input and output feature maps**: here $M$ denotes the number of input channels, $W$ and $D$ denote the spatial shape of the input feature maps, $K$ denotes the size of kernels, $W'$ and $D'$ are the spatial shape of the input feature maps, which depends on the padding.

$$s_{pqn}^{(k+1)} = \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{K}} \boldsymbol{w}_{ijn}^T \boldsymbol{x}_{p+i,q+j}^{(k)}, \ n = 1, \cdots, N, \tag{30}$$

where $\boldsymbol{x}_{pq}^{(k)}, \boldsymbol{s}_{pq}^{(k)} \in \mathbb{R}^M$, $\boldsymbol{s}_{pq}^{(k+1)} \in \mathbb{R}^N$ are the multi-channel pixels, and $s_{pqn}^{(k+1)}$ denotes the $n$-th channel of the pixel at $(p, q)$.

The multiplication between the scalars $x_{p+i,q+j}^{(k)}$ and $w_{ij}$ in Eq. (9) becomes the dot product between the column vectors $\boldsymbol{x}_{p+i,q+j}^{(k)}$ and $\boldsymbol{w}_{ijn}$. Here each $n$ corresponds to one output feature feature map. The forward-propagation process with multiple channels are illustrated in figure 2a and 2b.

When it comes to the back-propagation, we have

$$\boldsymbol{\delta}_{pq}^{(k)} = \frac{\partial e}{\partial \boldsymbol{s}_{pq}^{(k)}} = \theta'(\boldsymbol{s}_{pq}^{(k)}) \otimes \frac{\partial e}{\partial \boldsymbol{x}_{pq}^{(k)}} \tag{31}$$

$$= \theta'(\boldsymbol{s}_{pq}^{(k)}) \otimes \left( \sum_{n=1}^{N} \sum_{i \in \mathcal{K}, j \in \mathcal{K}} \frac{\partial e}{\partial s_{p+i,q+j,n}^{(k+1)}} \cdot \frac{\partial s_{p+i,q+j,n}^{(k+1)}}{\boldsymbol{x}_{pq}^{(k)}} \right) \tag{32}$$

$$= \theta'(\boldsymbol{s}_{pq}^{(k)}) \otimes \left( \sum_{n=1}^{N} \sum_{i \in \mathcal{K}, j \in \mathcal{K}} \frac{\partial e}{\partial s_{p+i,q+j,n}^{(k+1)}} \cdot \boldsymbol{w}_{-i,-j,n} \right) \tag{33}$$

$$= \theta'(s_{pq}^{(k)}) \otimes \left( \sum_{i \in \mathcal{K}, j \in \mathcal{K}} \sum_{n=1}^{N} \delta_{p+i,q+j,n}^{(k+1)} \cdot \boldsymbol{w}_{-i,-j,n} \right) \tag{34}$$

$$= \theta'(s_{pq}^{(k)}) \otimes \left( \sum_{i \in \mathcal{K}, j \in \mathcal{K}} \boldsymbol{W}_{-i,-j} \boldsymbol{\delta}_{p+i,q+j}^{(k+1)} \right) \tag{35}$$

where $\boldsymbol{W}_{-i,-j} = [\boldsymbol{w}_{-i,-j,1}, \boldsymbol{w}_{-i,-j,2}, \cdots, \boldsymbol{w}_{-i,-j,N}]$. From Eq. (32) to Eq. (33), we exchange the order of the two summations. From Eq. (33) to Eq. (34), we re-write the linear combination of vectors into a multiplication between a matrix (consists of the vectors) and a vector.

The formulas are very similar to the ones in the fully connected case and imply another connection between the convolution and the fully connected layer, which is, the channels in the two consecutive layers are fully connected to each other.

## Acknowledgements

We thank Zhengyang Wang for proof-reading. This work was supported in part by National Science Foundation grants IIS-1908220, IIS-1908198, IIS-1908166, DBI-1147134, DBI-1922969, DBI-1661289, CHE-1738305, National Institutes of Health grant 1R21NS102828, and Defense Advanced Research Projects Agency grant N66001-17-2-4031.

## References


[1] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. AML-Book New York, NY, USA:, 2012.

[2] Charu C Aggarwal. *Neural networks and deep learning*. Springer, 2018.