Backpropagation for Convolutional Neural Networks*

Dimitrios Katsaros Evangelia Fragkou Dimitrios Papakostas
Electrical & Computer Engineering department, University of Thessaly, Volos, Greece

Abstract

This short note intends to present the analytical calculations involved in computing backpropagation for convolutional neural networks, in particular for the convolutional and pooling layers.

Introduction

The emergence of deep learning [7] during the last decade has created a tremendous scientific/technological and also educational interest. Among the most popular deep neural networks are the convolutional neural networks (CNNs). They are considered as the state-of-the-art in image understanding after winning the image classification task in the ILSVRC competition for several years. In general, they are appropriate for processing grid-like input, such as images, video, time-series. Just like the traditional multi-layer perceptrons, CNNs are usually trained using backpropagation. Even though backpropagation is referred to as the standard training method for CNNs, however none of the popular or focused books on the topic [1, 2, 3, 4, 5, 6]provide sufficient or no details at all on how backpropagation equations are formulated for CNNs.

The goal of this short article is to fill this gap by exemplifying how backpropagation works in the case of one-dimensional input which is processed by a convolutional neural network. In particular, the article shows how gradients are calculated for the convolution and max-pooling layers.

Even though we assume that the reader is familiar with the architecture of a convolutional neural network, before proceeding to the next sections which show the backpropagation equations, we first show a small example of one-dimensional input, the output of a convolution filter applied to this input, and then the output of a max-pooling operation. Additionally, we show how neurons correspond to these operations. In order to keep things simple, we assume a zero bias term, and also we assume a pure linear activation function ($\sigma(x) = x$); therefore and concentrate on the particular features of a CNN, i.e., convolution and pooling layers. This basic architecture is depicted in Figure 1.

Convolution layer's gradients calculation

For the sake of simplicity, in the derivation of our equations we will use only a portion of the input presented in Figure 1, namely p_0–p_5. Let us start by computing the convolution of input $\mathbf{p} = [p_0, p_1, p_2, p_3, p_4, p_5]$ with the filter $\mathbf{w} = [w_0, w_1, w_2]$ using stride=1 and valid padding. Thus, we will get four outputs, namely z_0, z_1, z_2, z_3, one for each possible position of the filter $\mathbf{w}$ as it slides over the input $\mathbf{p}$. The equations defining the outputs will be the following (see

*How to cite: D. Katsaros, E. Fragkou, D. Papakostas, *Backpropagation for Convolutional Neural Networks*, Technical Report, Department of Electrical & Computer Engineering, University of Thessaly, 2021.

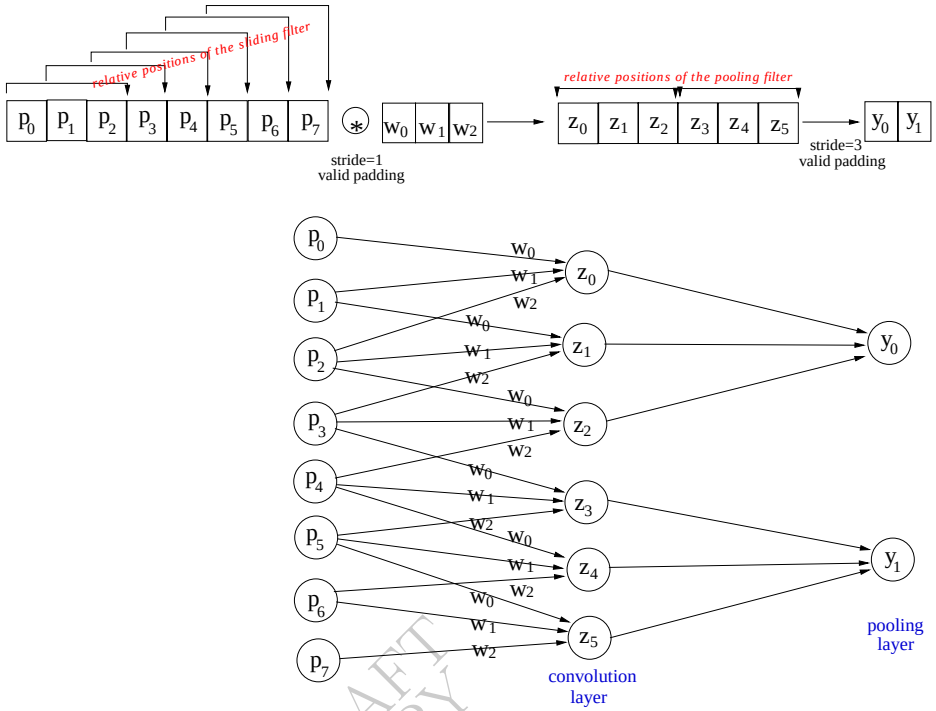

Figure 1. Part of a convolutional neural network applied to one-dimensional input, where an input layer is followed by a convolution layer and then by a max-pooling layer. The input need not be the actual input time-series, but any layer in a later processing stage of the CNN. Note that we omit the non-linearity layer, e.g., ReLU which may follow the convolution layer.

Figure 2):

$$z_0 = p_0 w_0 + p_1 w_1 + p_2 w_2 \tag{1}$$
$$z_1 = p_1 w_0 + p_2 w_1 + p_3 w_2 \tag{2}$$
$$z_2 = p_2 w_0 + p_3 w_1 + p_4 w_2 \tag{3}$$
$$z_3 = p_3 w_0 + p_4 w_1 + p_5 w_2 \tag{4}$$

From these four basic equation we will proceed to calculate the gradients of the loss function with respect to the filter $\mathbf{w}$ in the next subsection, and in the subsequent subsection we will calculate the gradients of the loss function with respect to the input $\mathbf{p}$.

Convolution layer's gradients calculation w.r.t. to a (any) filter

Let us compute the gradient $\nabla\hat{F}$ of $\hat{F}$ with respect to the filter $\mathbf{w} = [w_0, w_1, w_2]^T$, i.e., $\frac{\partial \hat{F}}{\partial \mathbf{w}} = [\frac{\partial \hat{F}}{\partial w_0}, \frac{\partial \hat{F}}{\partial w_1}, \frac{\partial \hat{F}}{\partial w_2}]^T$. We compute each partial derivative as follows:

$$\begin{aligned}\frac{\partial \hat{F}}{\partial w_0} &= \frac{\partial \hat{F}}{\partial z_0}\frac{\partial z_0}{\partial w_0} + \frac{\partial \hat{F}}{\partial z_1}\frac{\partial z_1}{\partial w_0} + \frac{\partial \hat{F}}{\partial z_2}\frac{\partial z_2}{\partial w_0} + \frac{\partial \hat{F}}{\partial z_3}\frac{\partial z_3}{\partial w_0} \Rightarrow \\ \frac{\partial \hat{F}}{\partial w_0} &= \frac{\partial \hat{F}}{\partial z_0}p_0 + \frac{\partial \hat{F}}{\partial z_1}p_1 + \frac{\partial \hat{F}}{\partial z_2}p_2 + \frac{\partial \hat{F}}{\partial z_3}p_3,\end{aligned} \tag{5}$$

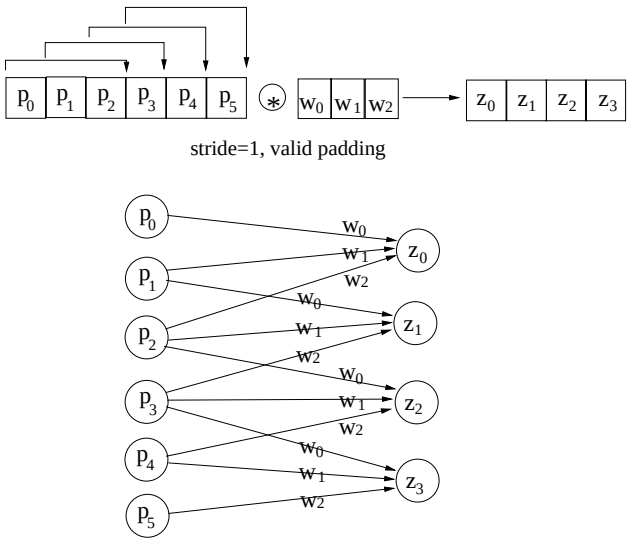

Figure 2. Convolution of a 1D input $\mathbf{p} = [p_0, p_1, p_2, p_3, p_4, p_5]$ with filter $\mathbf{w} = [w_0, w_1, w_2]$ with stride=1 and valid padding. Circles represent neurons, and the arcs represent connections among neurons.

and similarly, we get the following equations:

$$\frac{\partial \hat{F}}{\partial w_1} = \frac{\partial \hat{F}}{\partial z_0} p_1 + \frac{\partial \hat{F}}{\partial z_1} p_2 + \frac{\partial \hat{F}}{\partial z_2} p_3 + \frac{\partial \hat{F}}{\partial z_3} p_4, \tag{6}$$

$$\frac{\partial \hat{F}}{\partial w_2} = \frac{\partial \hat{F}}{\partial z_0} p_2 + \frac{\partial \hat{F}}{\partial z_1} p_3 + \frac{\partial \hat{F}}{\partial z_2} p_4 + \frac{\partial \hat{F}}{\partial z_3} p_5. \tag{7}$$

Now, suppose that we define the vector $\mathbf{s}$ as follows: $\mathbf{s} = [s_0, s_1, s_2, s_3] = [\frac{\partial \hat{F}}{\partial z_0}, \frac{\partial \hat{F}}{\partial z_1}, \frac{\partial \hat{F}}{\partial z_2}, \frac{\partial \hat{F}}{\partial z_3}]$ and let vector $\mathbf{p}$ be defined as follows: $\mathbf{p} = [p_0, p_1, p_2, p_3, p_4, p_5]$. If we convolve (stride=1, no padding) the two vectors, then we get the following vector:

$$\begin{aligned} \mathbf{p} \circledast \mathbf{s} = [&p_0 \frac{\partial \hat{F}}{\partial z_0} + p_1 \frac{\partial \hat{F}}{\partial z_1} + p_2 \frac{\partial \hat{F}}{\partial z_2} + p_3 \frac{\partial \hat{F}}{\partial z_3}, \\ &p_1 \frac{\partial \hat{F}}{\partial z_0} + p_2 \frac{\partial \hat{F}}{\partial z_1} + p_3 \frac{\partial \hat{F}}{\partial z_2} + p_4 \frac{\partial \hat{F}}{\partial z_3}, \quad p_2 \frac{\partial \hat{F}}{\partial z_0} + p_3 \frac{\partial \hat{F}}{\partial z_1} + p_4 \frac{\partial \hat{F}}{\partial z_2} + p_5 \frac{\partial \hat{F}}{\partial z_3}]. \end{aligned} \tag{8}$$

Therefore, we get the vector $\frac{\partial \hat{F}}{\partial \mathbf{w}} = [\frac{\partial \hat{F}}{\partial w_0}, \frac{\partial \hat{F}}{\partial w_1}, \frac{\partial \hat{F}}{\partial w_2}]$, and thus:

$$\frac{\partial \hat{F}}{\partial \mathbf{w}} = \mathbf{p} \circledast \mathbf{s}. \tag{9}$$

Conclusion #1. The gradient of the loss function with respect to a convolution filter is calculated by convolving the *deltas* (i.e., sensitivities) of the current layer with the inputs to the current layer.

Convolution layer's gradient calculation w.r.t. to its inputs

Now, let us compute the gradient $\nabla \hat{F}$ of $\hat{F}$ with respect to the inputs $\mathbf{p} = [p_0, p_1, p_2, p_3, p_4, p_5]$ to that layer. Starting again from Equations 1–4 and applying standard multivariate calculus

we get:

$$\frac{\partial \hat{F}}{\partial p_0} = \frac{\partial \hat{F}}{\partial z_0}\frac{\partial z_0}{\partial p_0} = s_0 w_0 \tag{10}$$

$$\frac{\partial \hat{F}}{\partial p_1} = \frac{\partial \hat{F}}{\partial z_0}\frac{\partial z_0}{\partial p_1} + \frac{\partial \hat{F}}{\partial z_1}\frac{\partial z_1}{\partial p_1} = s_0 w_1 + s_1 w_0 \tag{11}$$

$$\frac{\partial \hat{F}}{\partial p_2} = \frac{\partial \hat{F}}{\partial z_0}\frac{\partial z_0}{\partial p_2} + \frac{\partial \hat{F}}{\partial z_1}\frac{\partial z_1}{\partial p_2} + \frac{\partial \hat{F}}{\partial z_2}\frac{\partial z_2}{\partial p_2} = s_0 w_2 + s_1 w_1 + s_2 w_0 \tag{12}$$

$$\frac{\partial \hat{F}}{\partial p_3} = \frac{\partial \hat{F}}{\partial z_1}\frac{\partial z_1}{\partial p_3} + \frac{\partial \hat{F}}{\partial z_2}\frac{\partial z_2}{\partial p_3} + \frac{\partial \hat{F}}{\partial z_3}\frac{\partial z_3}{\partial p_3} = s_1 w_2 + s_2 w_1 + s_3 w_0 \tag{13}$$

$$\frac{\partial \hat{F}}{\partial p_4} = \frac{\partial \hat{F}}{\partial z_2}\frac{\partial z_2}{\partial p_4} + \frac{\partial \hat{F}}{\partial z_3}\frac{\partial z_3}{\partial p_4} = s_2 w_2 + s_3 w_1 \tag{14}$$

$$\frac{\partial \hat{F}}{\partial p_5} = \frac{\partial \hat{F}}{\partial z_3}\frac{\partial z_3}{\partial p_5} = s_3 w_2 \tag{15}$$

Consider again the vectors $\mathbf{s} = [s_0, s_1, s_2, s_3]$ and $\mathbf{w} = [w_0, w_1, w_2]$, but now *flip* vector $\mathbf{w}$ to get $flip(\mathbf{w}) = [w_2, w_1, w_0]$. Should we wish to calculate the convolution of $\mathbf{s}$ and $flip(\mathbf{w})$, then we should align them according to the relative positions shown in Figure 3.

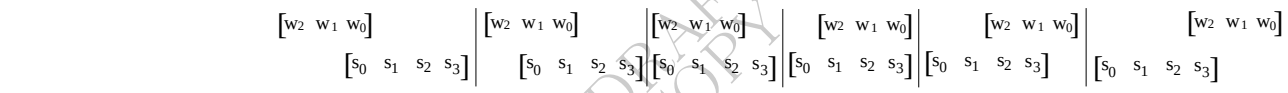

Figure 3. The six possible relative positions of the convolving vectors when full padding.

After performing the convolutions, we get the following vector:

$$\mathbf{s} \circledast flip(\mathbf{w}) = [s_0 w_0, \quad s_0 w_1 + s_1 w_0, \quad s_0 w_2 + s_1 w_1 + s_2 w_0, \\ s_1 w_2 + s_2 w_1 + s_3 w_0, \quad s_2 w_2 + s_3 w_1, \quad s_3 w_2]. \tag{16}$$

Comparing Equation 16 with the set of Equations 10–15, we deduce that the following relation holds true:

$$\mathbf{s} \circledast flip(\mathbf{w}) = [\frac{\partial \hat{F}}{\partial p_0}, \frac{\partial \hat{F}}{\partial p_1}, \frac{\partial \hat{F}}{\partial p_2}, \frac{\partial \hat{F}}{\partial p_3}, \frac{\partial \hat{F}}{\partial p_4}, \frac{\partial \hat{F}}{\partial p_5}] \Rightarrow \\ \frac{\partial \hat{F}}{\partial \mathbf{p}} = \mathbf{s} \circledast flip(\mathbf{w}). \tag{17}$$

Note that this convolution operation is performed with *full padding* the input.

Conclusion #2. The gradient of the loss function at a particular layer with respect to its input nodes of the previous layer is calculated by convolving the *deltas* (i.e., sensitivities) of the current layer with the flipped convolution filters [the convolution assumes full-padding].

Pooling layer's gradient calculation w.r.t. to its inputs

In this section we will calculate the gradients of the loss function in the max-pooling layer with respect to its inputs. We assume that our input consists of $\mathbf{z} = [z_0, z_1, z_2, z_3, z_4, z_5]$, and we

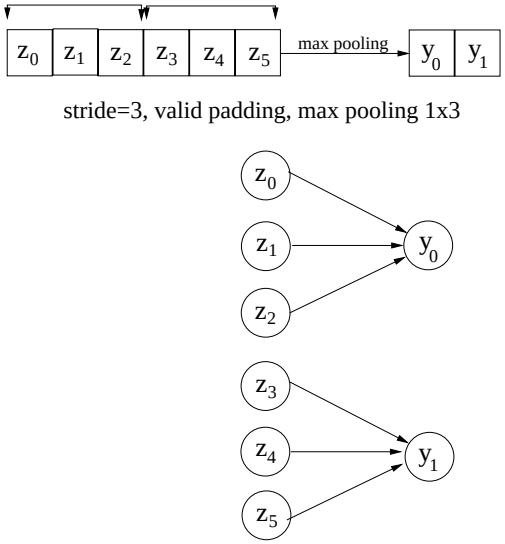

Figure 4. Max-pooling in a 1D input with stride=3, valid padding and pooling window 1×3.

apply a max-pooling operation with a sliding window of size 1×3, and *stride* equal to three; thus we have non-overlapping max-pooling operations. Moreover, we assume valid padding, i.e., no padding at all. Our output is thus: $y_0 = max\{z_0, z_1, z_2\}$ and $y_1 = max\{y_3, y_4, y_5\}$. In Figure 4 we can see the input and output neurons.

Then, the calculation of the gradients proceeds as follows:

$$\frac{\partial \hat{F}}{\partial z_0} = \frac{\partial \hat{F}}{\partial y_0}\frac{\partial y_0}{\partial z_0}, \quad \text{where } \frac{\partial y_0}{\partial z_0} = \begin{cases} 1, & \text{if } z_0 = max \\ 0, & \text{othewise} \end{cases} \tag{18}$$

$$\frac{\partial \hat{F}}{\partial z_1} = \frac{\partial \hat{F}}{\partial y_0}\frac{\partial y_0}{\partial z_1}, \quad \text{where } \frac{\partial y_0}{\partial z_1} = \begin{cases} 1, & \text{if } z_1 = max \\ 0, & \text{othewise} \end{cases} \tag{19}$$

$$\frac{\partial \hat{F}}{\partial z_2} = \frac{\partial \hat{F}}{\partial y_0}\frac{\partial y_0}{\partial z_2}, \quad \text{where } \frac{\partial y_0}{\partial z_2} = \begin{cases} 1, & \text{if } z_2 = max \\ 0, & \text{othewise} \end{cases} \tag{20}$$

$$\frac{\partial \hat{F}}{\partial z_3} = \frac{\partial \hat{F}}{\partial y_1}\frac{\partial y_1}{\partial z_3}, \quad \text{where } \frac{\partial y_1}{\partial z_3} = \begin{cases} 1, & \text{if } z_3 = max \\ 0, & \text{othewise} \end{cases} \tag{21}$$

$$\frac{\partial \hat{F}}{\partial z_4} = \frac{\partial \hat{F}}{\partial y_1}\frac{\partial y_1}{\partial z_4}, \quad \text{where } \frac{\partial y_1}{\partial z_4} = \begin{cases} 1, & \text{if } z_4 = max \\ 0, & \text{othewise} \end{cases} \tag{22}$$

$$\frac{\partial \hat{F}}{\partial z_5} = \frac{\partial \hat{F}}{\partial y_1}\frac{\partial y_1}{\partial z_5}, \quad \text{where } \frac{\partial y_1}{\partial z_5} = \begin{cases} 1, & \text{if } z_5 = max \\ 0, & \text{othewise} \end{cases} \tag{23}$$

Conclusion #3. The gradient of the previous layer is passed to neuron p_i, if and only if neuron p_i is the "winner" during the max-pooling operation, i.e., its value was selected as the outcome of the "max" operation.

Summary

In this short note we develop the backpropagation equations for a convolutional neural network appropriate for processing one-dimensional input. In particular, we show how the partial derivatives needed by backpropagation are computed in the convolution layer with respect to a (any) filter and with respect to the inputs, and also show the partial derivatives at the max-pooling layer with respect to its inputs.

References

[1] C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018.

[2] H. H Aghdam and E. J. Heravi. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. Springer, 2017.

[3] I. N. da Silva, D. H. Spatti, R. A. Flauzino, L. H. Bartocci-Liboni, and S. F. dos Reis Alves. *Artificial Neural Networks: A Practical Course*. Springer, 2017.

[4] K.-L. Du and M. N. S. Swamy. *Neural Networks and Statistical Learning*. Springer, 2014.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[6] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun. *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan-Claypool, 2018.

[7] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 251:436–444, 2015.