Backpropagation in Convolutional Neural Networks

University of Information Technology

Nguyễn Đặng Đức Mạnh

a sand soldier of 3ker

Table of Contents

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Geoffey Hinton

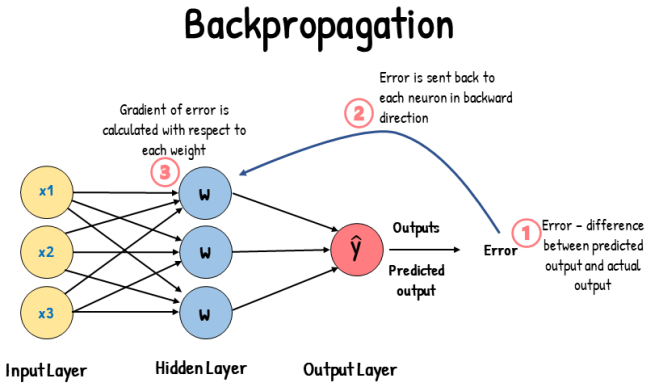
1 Introduction

Professor Geoffrey Hinton, the Nobel Prize winner in Physics in 2024, is also the one who popularized the backpropagation algorithm, which is the foundation for the remarkable development of AI

Backpropagation algorithm

1 Introduction

For a model m_θ parameterized by the parameter set θ, backpropagation is an algorithm that automatically updates the parameters θ to optimize m_θ based on the rules of derivatives..



Convolution

1 Introduction

The convolution operation has its origins in the field of signal processing. The kernel f acts as a signal detector; the more the signal g resembles f, the greater the response (output). A well-known application of convolution can be seen in the Fourier Transform.

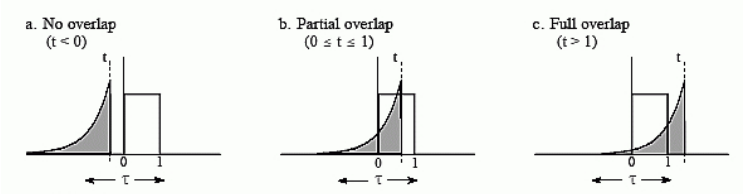
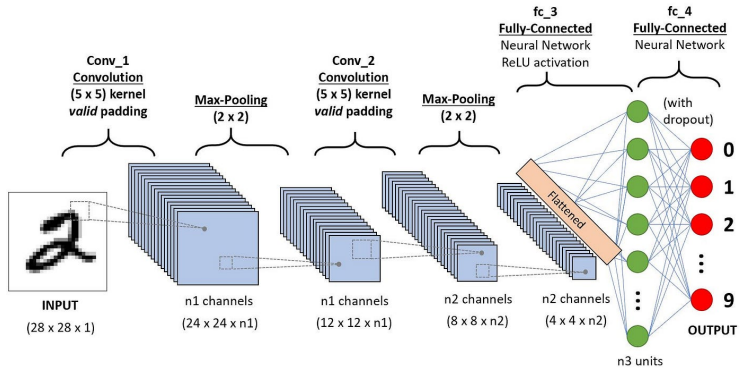


FIGURE 13-6
Calculating a convolution by segments. Since many continuous signals are defined by *regions*, the convolution calculation must be performed region-by-region. In this example, calculation of the output signal is broken into three sections: (a) no overlap, (b) partial overlap, and (c) total overlap, of the input signal and the shifted-flipped impulse response.

Convolutional Neural Network

1 Introduction

The convolution operation was subsequently applied alongside the backpropagation algorithm in computer vision tasks and began to become the dominant method to this day. The Artificial Neural Networks that use convolution for feature extraction are called Convolutional Neural Networks (CNNs).



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Objective

- Derive the formula for the convolution derivative.
- Develop the backpropagation process.
- Build and train a CNN based on the above theory.
- Validate the theory with experimental results.

Table of Contents

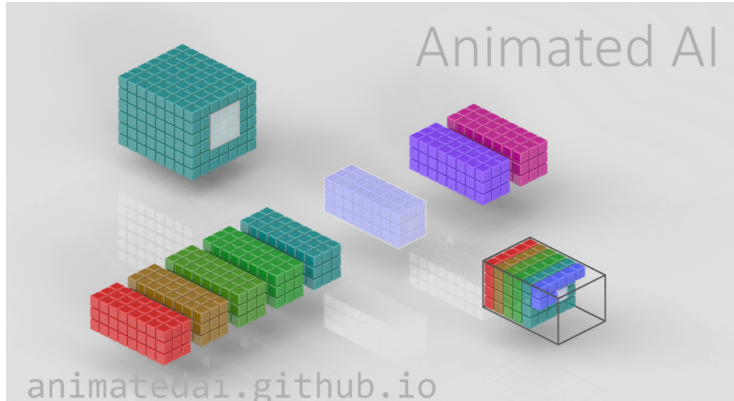
Section 2.1

Convolution

Misunderstanding

2 Theory

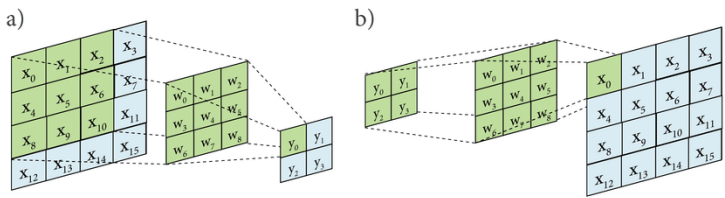
An RGB image has a shape of $[3, H, W]$, which means it is a 3-dimensional tensor. Similarly, a Conv2D operation has a kernel with 3 dimensions $[in\_channels, H_{conv}, W_{conv}]$, and there are $out\_channels$ such kernels, rather than being 2-dimensional as shown in many illustrative documents.



Misunderstanding

2 Theory

However, the $2D$ version is easier to explain, so in this slide, I will use $2D$ matrices to represent the kernel and the input/output feature maps. However, please note that in reality, both the kernel and the feature maps are $3D$ matrices (not counting the *batch_size*). In the implementation, I will still use the $3D$ version, which will be explained accordingly..



UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Convolution

2 Theory

In this slide, to avoid excessive explanations that may distract from the main focus, I would like to make the following assumptions:

- The kernel and feature map are square matrices.
- There is always a bias term.
- stride $= 1$, padding $= 1$, dilation $= 1$, groups $= 1$.

These assumptions may reduce generality, but they are very convenient for explaining and understanding convolution as well as its derivative. Once understood, generalizing is not difficult.

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

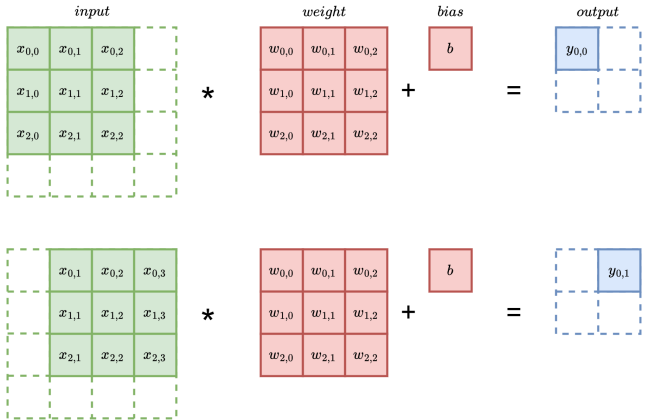
b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

Convolution

2 Theory



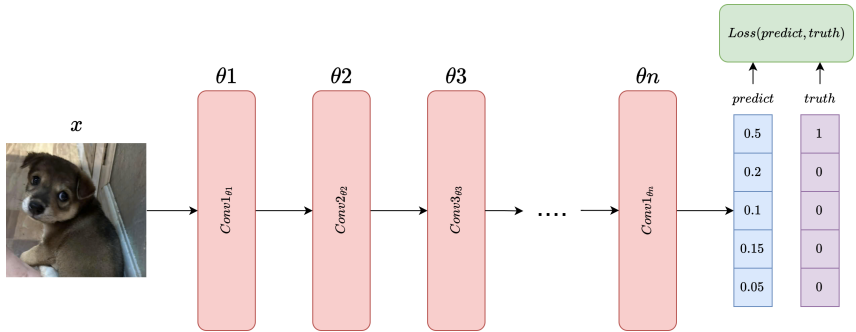
$$y_{0,0} = (x_{0,0} * w_{0,0} + x_{0,1} * w_{0,1} + x_{0,2} * w_{0,2} + x_{1,0} * w_{1,0} + x_{1,1} * w_{1,1} + x_{1,2} * w_{1,2} + x_{2,0} * w_{2,0} + x_{2,1} * w_{2,1} + x_{2,2} * w_{2,2}) + b$$

Section 2.2

Backpropagation

Backpropagation

2 Theory

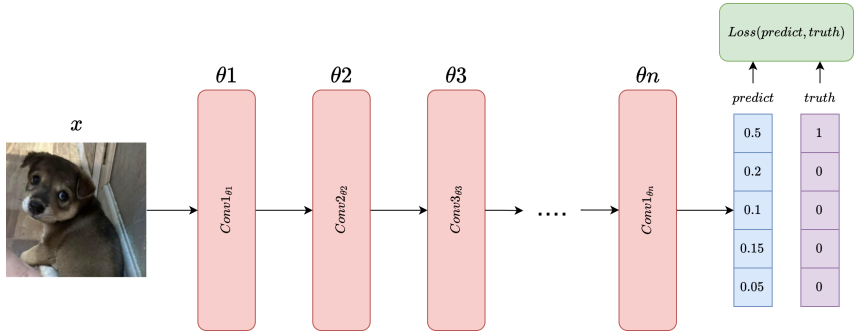


Suppose we need to build a CNN for the image classification task. The input is an image x, and there are n convolution layers $f_{\theta i}$, each parameterized by parameters θ_i. At this point, we can represent the output as:

$$\text{predict} = f_{\theta_n}(f_{\theta_{n-1}}(...(f_{\theta_1}(x))...)).$$

Backpropagation

2 Theory



Given a function Loss(predict, truth) that evaluates the fit between predict and truth, the smaller the Loss, the more accurate the model's predictions. We need to find the parameter set $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ that minimizes the Loss..

Backpropagation

2 Theory

At this point, you may realize that we only need to compute the generalized derivative and find the extrema of the Loss function with respect to θ. Unfortunately, this is practically infeasible due to the high dimensionality and complexity of the function. Although we cannot find the generalized derivative, for each input x, we can compute the derivative of Loss with respect to θ at x. By moving in the direction opposite to the derivative, we will approach a local extremum of the Loss.

$$\frac{\partial f(g(x))}{\partial x} = \frac{\partial f(g(x))}{\partial g(x)} \frac{\partial g(x)}{\partial x}$$

Since $\text{predict} = f_{\theta_n}(f_{\theta_{n-1}}(...(f_{\theta_1}(x))...))$ is a composite function, we can apply the chain rule above to calculate the derivative of Loss with respect to each θ_i one by one. The first derivative is taken at the output and then traced back to the input, which is why it is called backpropagation.

Section 2.3

Derivative of Weight

Derivative of Weight

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

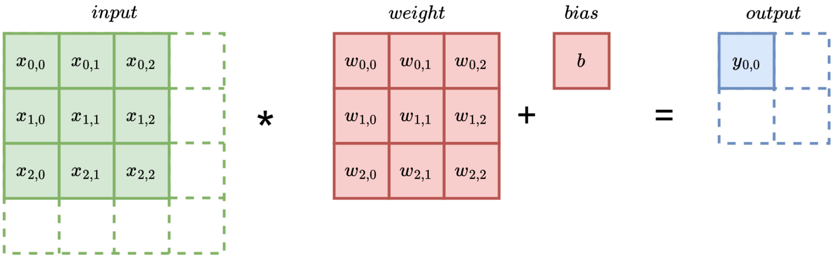
b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

$$\frac{\partial Loss(x)}{\partial weight} = \frac{\partial Loss(x)}{\partial output}\frac{\partial output}{\partial weight}$$

Derivative of Weight



$$y_{0,0} = (x_{0,0} * \color{red}{w_{0,0}} + x_{0,1} * w_{0,1} + x_{0,2} * w_{0,2} + x_{1,0} * w_{1,0} + x_{1,1} * w_{1,1} + x_{1,2} * w_{1,2} + x_{2,0} * w_{2,0} + x_{2,1} * w_{2,1} + x_{2,2} * w_{2,2}) + b$$

$$y_{0,0} = (x_{0,0} * w_{0,0} + x_{0,1} * \color{red}{w_{0,1}} + x_{0,2} * w_{0,2} + x_{1,0} * w_{1,0} + x_{1,1} * w_{1,1} + x_{1,2} * w_{1,2} + x_{2,0} * w_{2,0} + x_{2,1} * w_{2,1} + x_{2,2} * w_{2,2}) + b$$

$$y_{0,0} = (x_{0,0} * w_{0,0} + x_{0,1} * w_{0,1} + x_{0,2} * \color{red}{w_{0,2}} + x_{1,0} * w_{1,0} + x_{1,1} * w_{1,1} + x_{1,2} * w_{1,2} + x_{2,0} * w_{2,0} + x_{2,1} * w_{2,1} + x_{2,2} * w_{2,2}) + b$$

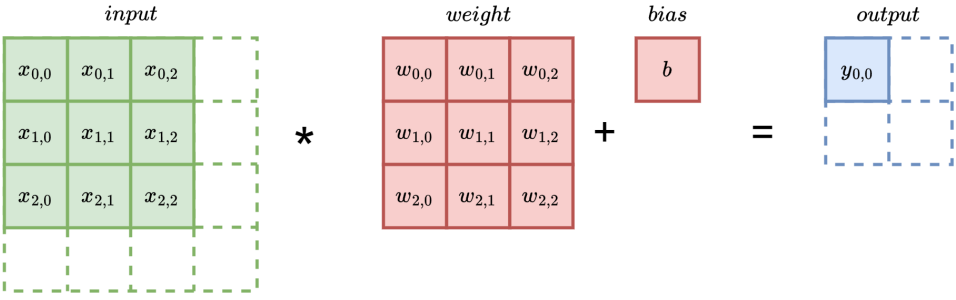
$$y_{0,0} = (x_{0,0} * w_{0,0} + x_{0,1} * w_{0,1} + x_{0,2} * w_{0,2} + x_{1,0} * \color{red}{w_{1,0}} + x_{1,1} * w_{1,1} + x_{1,2} * w_{1,2} + x_{2,0} * w_{2,0} + x_{2,1} * w_{2,1} + x_{2,2} * w_{2,2}) + b$$

$$y_{0,0} = (x_{0,0} * w_{0,0} + x_{0,1} * w_{0,1} + x_{0,2} * w_{0,2} + x_{1,0} * w_{1,0} + x_{1,1} * \color{red}{w_{1,1}} + x_{1,2} * w_{1,2} + x_{2,0} * w_{2,0} + x_{2,1} * w_{2,1} + x_{2,2} * w_{2,2}) + b$$

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of Weight

2 Theory

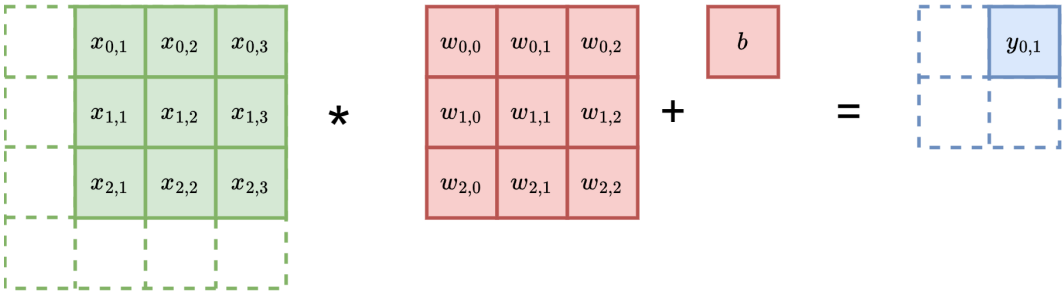


$$\frac{\partial y_{0,0}}{\partial weight} = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} \\ x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \end{bmatrix}$$

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of Weight

2 Theory



$$\frac{\partial y_{0,1}}{\partial weight} =$$

$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$

Derivative of Weight

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

$$\frac{\partial Loss}{\partial weight} = \frac{\partial Loss}{\partial y_{0,0}} \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} \\ x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \end{bmatrix} + \frac{\partial Loss}{\partial y_{0,1}} \begin{bmatrix} x_{0,1} & x_{0,2} & x_{0,3} \\ x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{bmatrix} + \frac{\partial Loss}{\partial y_{1,0}} \begin{bmatrix} x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \\ x_{3,0} & x_{3,1} & x_{3,2} \end{bmatrix} + \frac{\partial Loss}{\partial y_{1,1}} \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix}$$

Section 2.4

Derivative of Bias

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of Convolution

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

$$
\begin{aligned}
y_{0,0} &= (\dots) + b \\
y_{0,1} &= (\dots) + b \\
y_{1,0} &= (\dots) + b \\
y_{1,1} &= (\dots) + b
\end{aligned}
$$

Derivative of bias

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

$$\frac{\partial Loss}{\partial bias} = \frac{\partial Loss}{\partial y_{0,0}} \boxed{1} + \frac{\partial Loss}{\partial y_{0,1}} \boxed{1} + \frac{\partial Loss}{\partial y_{1,0}} \boxed{1} + \frac{\partial Loss}{\partial y_{1,1}} \boxed{1}$$

Section 2.5

Derivative of Input

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of Input

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

$$\frac{\partial y_{0,0}}{\partial input} =$$

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$	0
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$	0
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$	0
0	0	0	0

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of Input

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

$$\frac{\partial y_{0,1}}{\partial input} = \begin{matrix} 0 & w_{0,0} & w_{0,1} & w_{0,2} \\ 0 & w_{1,0} & w_{1,1} & w_{1,2} \\ 0 & w_{2,0} & w_{2,1} & w_{2,2} \\ 0 & 0 & 0 & 0 \end{matrix}$$

Derivative of Input

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Kernel weight

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

Kernel bias

b

Output

$y_{0,0}$	$y_{0,1}$
$y_{1,0}$	$y_{1,1}$

$$\frac{\partial Loss}{\partial input} = \frac{\partial Loss}{\partial y_{0,0}} \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 \\ w_{1,0} & w_{1,1} & w_{1,2} & 0 \\ w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{\partial Loss}{\partial y_{0,1}} \begin{bmatrix} 0 & w_{0,0} & w_{0,1} & w_{0,2} \\ 0 & w_{1,0} & w_{1,1} & w_{1,2} \\ 0 & w_{2,0} & w_{2,1} & w_{2,2} \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{\partial Loss}{\partial y_{1,0}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ w_{0,0} & w_{0,1} & w_{0,2} & 0 \\ w_{1,0} & w_{1,1} & w_{1,2} & 0 \\ w_{2,0} & w_{2,1} & w_{2,2} & 0 \end{bmatrix} + \frac{\partial Loss}{\partial y_{1,1}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} \\ 0 & w_{1,0} & w_{1,1} & w_{1,2} \\ 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix}$$

Section 2.6

Derivative of Padding

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of Padding

2 Theory

Input

$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Padding

Output

0	0	0	0	0	0
0	$y_{0,0}$	$y_{0,1}$	$y_{0,2}$	$y_{0,3}$	0
0	$y_{1,0}$	$y_{1,1}$	$y_{1,2}$	$y_{1,3}$	0
0	$y_{2,0}$	$y_{2,1}$	$y_{2,2}$	$y_{2,3}$	0
0	$y_{3,0}$	$y_{3,1}$	$y_{3,2}$	$y_{3,3}$	0
0	0	0	0	0	0

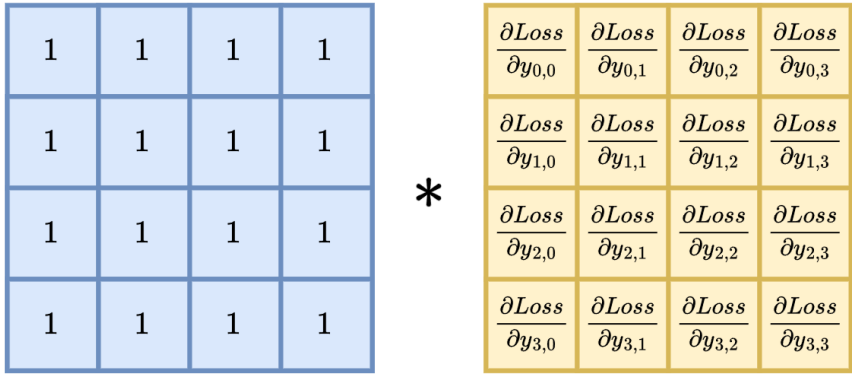
$$\frac{\partial y_{i,j}}{\partial x_{k,l}} = 1 \text{ when } i = k, j = l$$

$$\frac{\partial y_{i,j}}{\partial x_{k,l}} = 0 \text{ otherwise}$$

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Derivative of Padding

2 Theory

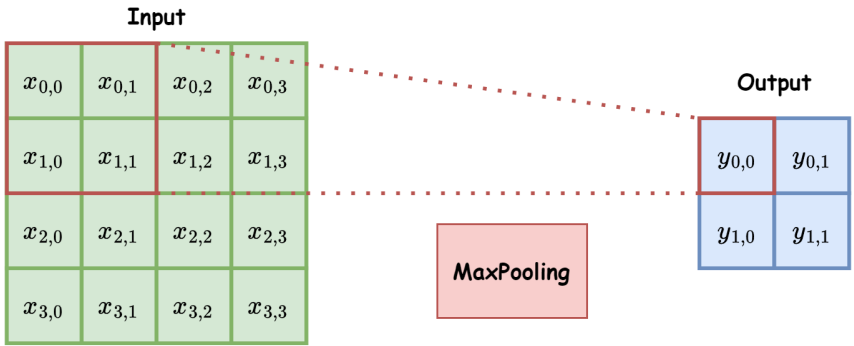


Section 2.7

Derivative of MaxPool

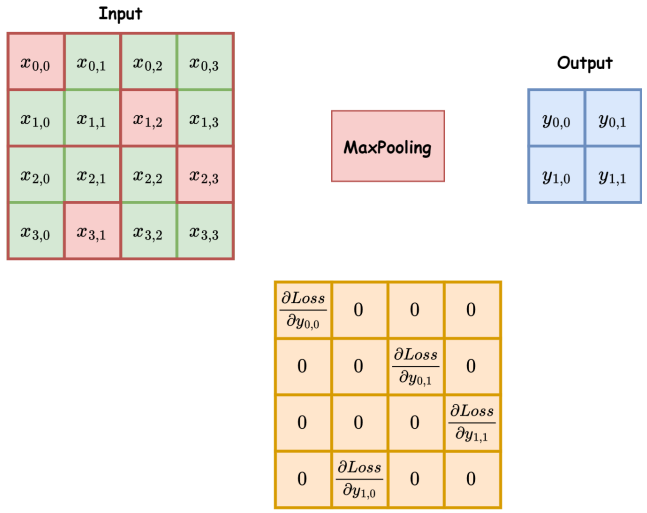
Derivative of MaxPool

2 Theory



Derivative of MaxPool

2 Theory



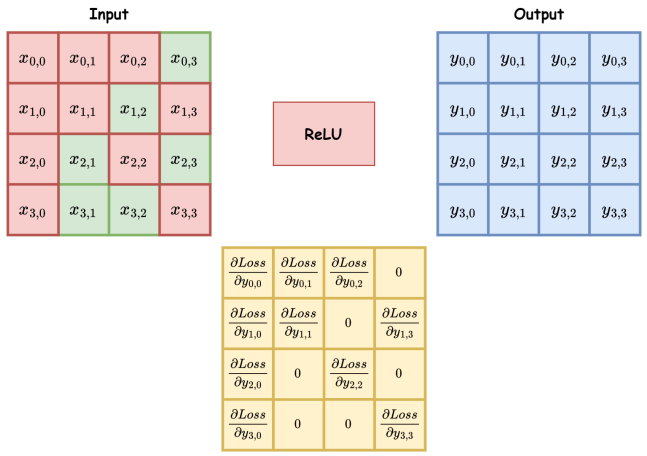
Section 2.8

Derivative of ReLU

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of ReLU

2 Theory



$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$
$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

$y_{0,0}$	$y_{0,1}$	$y_{0,2}$	$y_{0,3}$
$y_{1,0}$	$y_{1,1}$	$y_{1,2}$	$y_{1,3}$
$y_{2,0}$	$y_{2,1}$	$y_{2,2}$	$y_{2,3}$
$y_{3,0}$	$y_{3,1}$	$y_{3,2}$	$y_{3,3}$

$\frac{\partial Loss}{\partial y_{0,0}}$	$\frac{\partial Loss}{\partial y_{0,1}}$	$\frac{\partial Loss}{\partial y_{0,2}}$	0
$\frac{\partial Loss}{\partial y_{1,0}}$	$\frac{\partial Loss}{\partial y_{1,1}}$	0	$\frac{\partial Loss}{\partial y_{1,3}}$
$\frac{\partial Loss}{\partial y_{2,0}}$	0	$\frac{\partial Loss}{\partial y_{2,2}}$	0
$\frac{\partial Loss}{\partial y_{3,0}}$	0	0	$\frac{\partial Loss}{\partial y_{3,3}}$

Section 2.9

Derivative of Cross Entropy

UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Derivative of Cross Entropy

2 Theory

The problem we set out to solve is image classification, where the output predict $\in \mathbb{R}^N$ with predict$[i]$ being the prediction for class i, and the label truth $\in \mathbb{R}^n$ (abbreviated as t) is a one-hot vector. We have the Softmax function:

$$p_i = \frac{e^{\text{predict}[i]}}{\sum_{j=1}^{N} e^{\text{predict}[j]}}$$

The cross-entropy loss function is defined as (using natural logarithm):

$$\text{Loss} = -\sum_{i=1}^{N} t[i] \log(p[i])$$

Derivative of Cross Entropy

2 Theory

However, the vector t is one-hot, with only one value being 1 and the rest being 0. Let k be the index where $t[k] = 1$. Thus, the loss function can be simplified to:

$$\text{Loss} = -\log(p[k])$$

Let's recall some derivatives of composite functions u and v:

$$\begin{aligned}
(\log(u))' &= \frac{u'}{u} \\
\left(\frac{1}{u}\right)' &= \frac{-u'}{u^2} \\
(uv)' &= u'v + uv'
\end{aligned}$$

Derivative of Cross Entropy

2 Theory

We need to calculate $\frac{\partial \text{Loss}}{\partial \text{predict}}$ (from now on, we will refer to predict as x): For $x[i]$ where $i = k$:

$$\begin{aligned}
\frac{\partial Loss}{\partial x[i]} &= -\frac{\partial log(p[k])}{\partial x[i]} \\
&= -\frac{1}{p[k]} \frac{\partial p[k]}{\partial x[i]} \\
&= -\frac{1}{p[k]} \left(\frac{e^{x[k]}}{\sum_{j=1}^{N} e^{x[j]}} - \left(\frac{e^{x[k]}}{\sum_{j=1}^{N} e^{x[j]}}\right)^2\right) \\
&= -\frac{1}{p[k]} (p[k] - p[k]^2) \\
&= -(1 - p[k]) \\
&= p[k] - 1
\end{aligned}$$

Derivative of Cross Entropy

2 Theory

For $x[i]$ where $i \neq k$:

$$\begin{aligned}
\frac{\partial Loss}{\partial x[i]} &= -\frac{\partial log(p[k])}{\partial x[i]} \\
&= -\frac{1}{p[k]}\frac{\partial p[k]}{\partial x[i]} \\
&= -\frac{1}{p[k]}(-\frac{e^{x[i]}e^{x[k]}}{(\sum_{j=1}^{N} e^{x[j]})^2}) \\
&= \frac{1}{p[k]}(p[k]p[i]) \\
&= p[i]
\end{aligned}$$

Table of Contents

Section 3.1

Setting

Model

3 Experiment

Based on the above theory, I built a model using Conv2D, ReLU, Softmax, and the Cross Entropy loss function entirely with NumPy (not using PyTorch's autograd in the model, only for the purpose of checking consistency of results) and manually computed derivatives. The code is attached in the document. The model consists of 6 Conv layers:

```
    def __init__(self,
input_dim, eta=1e-4):
    self.input_dim = input_dim
    self.eta = eta

    self.conv1    = Conv2D(1,
16, 3)
    self.padding1 = Padding(1)


    self.relu1    = ReLU()

    self.conv2    = Conv2D(16,
 32, 3)
    self.conv3    = Conv2D(32,
64, 3)
    self.relu3    = ReLU()


    self.conv4    = Conv2D(64,
64, 3)
    self.relu4    = ReLU()

    self.flatten  = Flatten()

    self.conv5    = Conv2D
(64*10*10, 10, 1)
```

MNIST

3 Experiment

MNIST is a dataset for handwritten digit classification consisting of 60,000 training images and 10,000 test images (28x28 pixels).

Section 3.2

Kết quả trực quan

Visualize

3 Experiment

Visualize time!

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Q&A

Thank you for listening!
Your feedback will be highly appreciated!