



Mathematical morphology meets Deep Learning

Santiago Velasco-Forero, Samy Blusseau, Mateus Sangalli

► To cite this version:

Santiago Velasco-Forero, Samy Blusseau, Mateus Sangalli. Mathematical morphology meets Deep Learning. 13th European Congress for Stereology and Image Analysis (ECSIA), Jun 2021, Saint-Étienne, France. hal-03355356

HAL Id: hal-03355356

<https://hal.science/hal-03355356>

Submitted on 27 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mathematical morphology meets Deep Learning

ECSIA mini-course

Santiago Velasco-Forero, Mateus Sangalli, Samy Blusseau

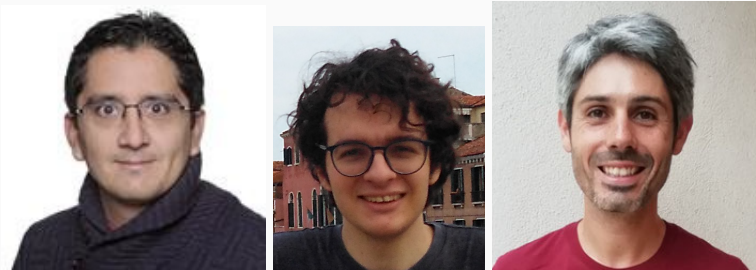
Center for Mathematical Morphology - Mines ParisTech - PSL Research University, France

<https://github.com/Jacobiano/morpholayers>

Table of contents

1. Introduction
2. Deep Learning in 15 minutes
3. Mathematical morphology - Learning simple translation invariant operators
4. Depthwise Morphological Layers
5. Morphological Scale-Spaces: Introduction to Invariance and Equivariance

Introduction



Left to right: Santiago, Mateus and Samy

Mini-Course goals:

- Study some interactions between mathematical morphology and Deep Learning.
- Make our code accessible for everyone.

<https://github.com/Jacobiano/morpholayers>

Course Description:

- A quick introduction to Deep Learning (15 minutes) (14:05 - 14:20) (S. Velasco-Forero)
- An introduction to Mathematical Morphology and learning operators (65 minutes) (14:20 - 15:30) (S. Blusseau)
- Pause (15:30 - 15:45)
- Depthwise Morphological Layers (45 minutes) (15:45 - 16:30) (S. Velasco-Forero)
- Morphological Scale-Spaces (45 minutes) (16:30 - 17:15) (M. Sangalli)
- Course link: [*https://github.com/Jacobiano/morpholayers*](https://github.com/Jacobiano/morpholayers)
- Seven online tutorials.

Deep Learning in 15 minutes

Learning from data

- Learning from data consists in using examples $\{(\mathbf{x}_k, \mathbf{y}_k)\}_{1 \leq k \leq n} \in (\mathcal{X}, \mathcal{Y})$ to build a **parametric map** $\phi : \mathcal{X} \mapsto \mathcal{Y}$ that accurately predicts the value \mathbf{y}_{n+1} for any new sample \mathbf{x}_{n+1} , that is $\mathbf{y}_{n+1} \approx \phi(\mathbf{x}_{n+1})$

Learning from data

- Learning from data consists in using examples $\{(\mathbf{x}_k, \mathbf{y}_k)\}_{1 \leq k \leq n} \in (\mathcal{X}, \mathcal{Y})$ to build a **parametric map** $\phi : \mathcal{X} \mapsto \mathcal{Y}$ that accurately predicts the value \mathbf{y}_{n+1} for any new sample \mathbf{x}_{n+1} , that is $\mathbf{y}_{n+1} \approx \phi(\mathbf{x}_{n+1})$

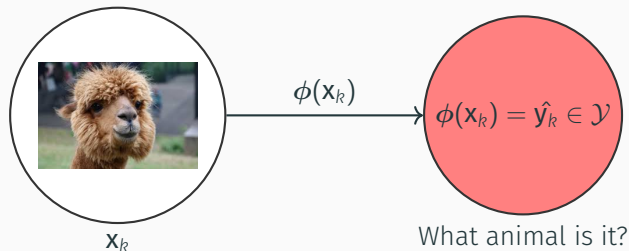


Figure 1: Supervised classification

The pairs $(\mathbf{x}_k, \mathbf{y}_k)$ define the nature of the problem, supervised, denoising, segmentation, image transformation, and so on.

- A Deep Neural Network with d layers and parameters θ , in its simplest form, is a *compositional map* that may be written as:

$$\phi_{f;\theta}(\mathbf{x}) : f^{(d)} \circ f^{(d-1)} \circ f^{(d-2)} \dots \circ f^1(\mathbf{x})$$

Usually, we use the term *deep* when d (number of layers) is larger than two.

Some references:

1. Deep Learning. <https://www.deeplearningbook.org/>
2. Dive into Deep Learning. <https://d2l.ai/>

Dense Layer

- Dense layer , $f^{(i)}(\mathbf{x}) := g^{(i)}(\mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)})$,

with $g^{(i)}$ is a non-linear function (activation), and $\theta = [\mathbf{W}^{(i)}, \mathbf{b}^{(i)}]$ (parameters)

- The weights $\mathbf{W}_{p \times k}^{(i)}$ are multiplied with the inputs variables.
- The bias $\mathbf{b}_{k \times 1}^{(i)}$ can be interpreted as a threshold on the sum.
- The activation function somehow decides, depending on its input, if a signal (the neuron's activation) is produced.
- A dense layer is a parametric mapping $\mathbb{R}^p \mapsto \mathbb{R}^k$, where k is called the number of units.

Loss function / Empirical Risk

- In the training process, we would usually tune the parameters θ so as to minimise the difference between the labels (ideal maps $\{\mathbf{y}_k\}$) and the outputs of the DNN (estimated maps $\{\hat{\mathbf{y}}_k := \phi_{f;\theta}(\mathbf{x}_k)\}$) at any training instance \mathbf{x}_k , such that the difference goes to zero as the number of samples n increases.
- For that, we define a loss function denoted by $Loss : (\mathcal{Y} \times \mathcal{Y}) \mapsto \mathbb{R}^+$, represents the difference between the labels and the DNN output.
- Usually, the training process, minimise the called **empirical risk**, by averaging the loss function on a large set of training examples $\{(\mathbf{x}_k, \mathbf{y}_k)\}$

$$\hat{\theta} := \arg \min \sum_{i=1}^n Loss(\mathbf{y}_k, \phi_{f;\theta}(\mathbf{x}_k)) \quad (1)$$

Minimization by backpropagation and SGD

This minimization is usually done via **stochastic gradient descend** (SGD).

1. SGD starts from certain initial θ (**Initialization**).
2. Compute the loss function for some input examples (\mathbf{x}). (**Forward Pass**)
3. The computation of gradient with respect to the loss function via the chain rule in networks, denoted by $\nabla_{\theta} \text{Loss}(\mathbf{x})$. (**Back-propagation**)
4. Updates each parameter by moving it in the direction of the negative gradient, $\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} \text{Loss}(\mathbf{x})$, where η is called the learning rate. (**GD update**)

The term *stochastic* in SGD indicates that a random small number of training samples, called a **batch** is used in the gradient calculation. A pass of the whole training set is called an **epoch**. Usually, after each epoch, the error on a validation dataset is evaluated and when it stabilizes the training is complete.

Backpropagation applied to neural networks

In the case of neural networks, the loss function depends on each parameter θ_i via the composition of several simple functions. In order to compute the gradient $\nabla_{\theta} \text{Loss}(\mathbf{x})$ we will make extensive use of the vector chain rule, i.e., $\frac{\partial}{\partial \mathbf{x}} \mathbf{f}(\mathbf{g}(\mathbf{x})) = \frac{\partial \mathbf{f}}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$.

$$\nabla_{\theta} \text{Loss}(\mathbf{y}_k, \phi_{f,\theta}(\mathbf{x}_k)) = \frac{\partial \text{Loss}}{\partial \phi_{f,\theta}(\mathbf{x}_k)} \frac{\partial \phi_{f,\theta}(\mathbf{x})}{\partial \theta} \quad (2)$$

Backpropagation applied to neural networks

In the case of neural networks, the loss function depends on each parameter θ_i via the composition of several simple functions. In order to compute the gradient $\nabla_{\theta} \text{Loss}(\mathbf{x})$ we will make extensive use of the vector chain rule, i.e., $\frac{\partial}{\partial \mathbf{x}} \mathbf{f}(\mathbf{g}(\mathbf{x})) = \frac{\partial \mathbf{f}}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$.

$$\nabla_{\theta} \text{Loss}(\mathbf{y}_k, \phi_{f,\theta}(\mathbf{x}_k)) = \frac{\partial \text{Loss}}{\partial \phi_{f,\theta}(\mathbf{x}_k)} \frac{\partial \phi_{f,\theta}(\mathbf{x})}{\partial \theta} \quad (2)$$

In DL frameworks as Tensorflow, Pytorch, MxNet the computation of the gradient is performed by Automatic Differentiation

Gradient Descent: the intuition [Cauchy, 1847]

Given a dataset \mathbf{X} and a model with parameters $\boldsymbol{\theta}$, we would like to find the best values for $\boldsymbol{\theta}$ such that minimize a given loss function $Loss$ evaluated on \mathbf{X} .

Algorithm 1 pseudocode gradient descent

- 1: **given** initial learning rate $\eta \in \mathbb{R}$ and dataset \mathbf{X}
 - 2: **initialize** time step $t = 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^P$,
 - 3: **repeat**
 - 4: $t=t+1$
 - 5: $\mathbf{g}_t = \nabla Loss_i(\mathbf{X}, \boldsymbol{\theta}_{t-1})$ *Return gradient via backpropagation*
 - 6: $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \mathbf{g}_t$
 - 7: **until** stopping criterion is met
 - 8: **return** optimized parameters $\boldsymbol{\theta}_i$
-

We note that in this first version, the gradient is calculated for the entire training set.

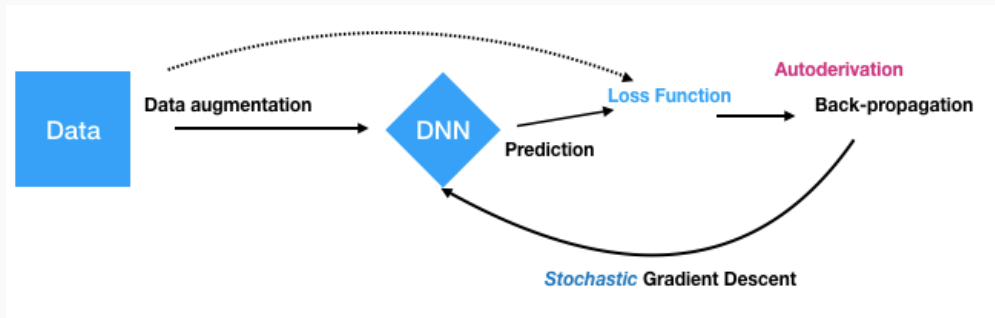
Stochastic Gradient Descent [Robbins and Monro, 1985]

Algorithm 2 pseudocode for stochastic gradient descent

- 1: **given** initial learning rate $\eta \in \mathbb{R}$ and dataset X
 - 2: **initialize** time step $t = 0$, parameter vector $\theta_{t=0} \in \mathbb{R}^P$,
 - 3: **repeat**
 - 4: $t=t+1$
 - 5: $X_t = \text{SelectBatch}(X)$ *Select a batch from data, whole data, only one, ...*
 - 6: $g_t = \nabla \text{Loss}_i(X_t, \theta_{t-1})$ *Return gradient via backpropagation*
 - 7: $\theta_t = \theta_{t-1} - \eta g_t$
 - 8: **until** stopping criterion is met
 - 9: **return** optimized parameters θ_i
-

Here, we draw m samples to calculate the gradient.

Summary Deep Learning



Tutorial 0: Deep Learning in 15 minutes.

<https://github.com/Jacobiano/morpholayers>

Mathematical morphology - Learning simple translation invariant operators

Mathematical morphology - Learning simple translation invariant operators

Introduction to translation invariant
morphological operators

Analogy with linear image processing



Original



Gaussian blur



Dilation

Analogy with linear image processing



Original



Gaussian blur



Erosion

Analogy with linear image processing

In this course:

Mathematical Morphology is seen as a
non-linear counterpart of linear image processing.

The set of images of size $M \times N$ pixels can be identified to a vector space
(e.g. $\mathbb{R}^{M \times N}$, periodic functions mapping \mathbb{R}^2 to \mathbb{R} or \mathbb{R}^3 , or mapping \mathbb{Z}^2 to \mathbb{R} or \mathbb{R}^3)

But this is an approximation:

in practice, pixel values are non-negative and quantized (e.g. integer values in $[0, 255]$ for 8-bits images)

It can also be identified to a complete lattice
(and this is not an approximation!)

Analogy with linear image processing

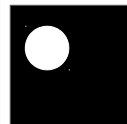
Complete lattice

A partially ordered set (L, \leq) is a complete lattice if every subset $A \subseteq L$ has

- a least upper bound (the *supremum*), denoted $\bigvee A$,
- a greatest lower bound (the *infimum*), denoted $\bigwedge A$.

Example 1: $(\mathcal{P}(E), \subseteq)$ for any set E

- $\bigvee \mathcal{P}(E) = E$
- $\bigwedge \mathcal{P}(E) = \emptyset$
- For any collection $\mathcal{C} = (A_i)_{i \in I}$ of subsets of E ,
 $\bigvee \mathcal{C} = \bigcup_{i \in I} A_i$ and $\bigwedge \mathcal{C} = \bigcap_{i \in I} A_i$.



$A \subset E$



$A \vee B = A \cup B$



$B \subset E$



$A \wedge B = A \cap B$

Analogy with linear image processing

Complete lattice

A partially ordered set (L, \leq) is a complete lattice if every subset $A \subseteq L$ has

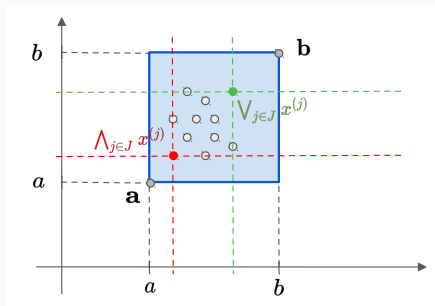
- a least upper bound (the *supremum*), denoted $\bigvee A$,
- a greatest lower bound (the *infimum*), denoted $\bigwedge A$.

Example 2: $([a, b]^n, \subseteq_n)$, with $a \leq b \in \mathbb{R}$, $n \in \mathbb{N}^*$ and $\forall x, y \in [a, b]^n$, $x \leq_n y \iff x_i \leq y_i \ \forall i \in \{1, \dots, n\}$

- $\bigvee [a, b]^n = [b \dots b]^T = \mathbf{b}$
- $\bigwedge [a, b]^n = [a \dots a]^T = \mathbf{a}$
- $\bigvee_{j \in J} x^{(j)} = [\bigvee_j x_1^{(j)}, \dots, \bigvee_j x_n^{(j)}]$
- $\bigwedge_{j \in J} x^{(j)} = [\bigwedge_j x_1^{(j)}, \dots, \bigwedge_j x_n^{(j)}]$

Typically, 8-bits images:

$a = 0$, $b = 255$ and n the number of pixels.



Vector spaces

- Closed under **linear combinations**
- Mappings commuting with linear combinations are the **linear applications**

$$f\left(\sum_i \lambda_i x_i\right) = \sum_i \lambda_i f(x_i)$$

Complete lattices

- Closed under **supremum and infimum**
- Mappings commuting with the supremum are the **dilations**

$$\delta\left(\bigvee_i x_i\right) = \bigvee_i \delta(x_i)$$

- Mappings commuting with the infimum are the **erosions**.

$$\varepsilon\left(\bigwedge_i x_i\right) = \bigwedge_i \varepsilon(x_i)$$

Analogy with linear image processing

Let \mathcal{F} be the set of functions mapping \mathbb{Z}^2 to \mathbb{R} , and $L : \mathcal{F} \rightarrow \mathcal{F}$ an operator.

Translation invariance

For any $u \in \mathbb{Z}^2$, the translation $\tau_u : \mathcal{F} \rightarrow \mathcal{F}$ is the operator defined by: $\tau_u f : x \mapsto f(x - u)$.

We say that L is translation-invariant if for any $u \in \mathbb{Z}^2$, $L \circ \tau_u = \tau_u \circ L$.



Linear convolution (Riesz representation theorem)

The operator L is *linear and translation-invariant* if and only if there exists $h \in \mathcal{F}$ such that for any $f \in \mathcal{F}$

$$\forall x \in \mathbb{Z}^2, Lf(x) = f * h(x) = \sum_{y \in \mathbb{Z}^2} f(y)h(x - y).$$

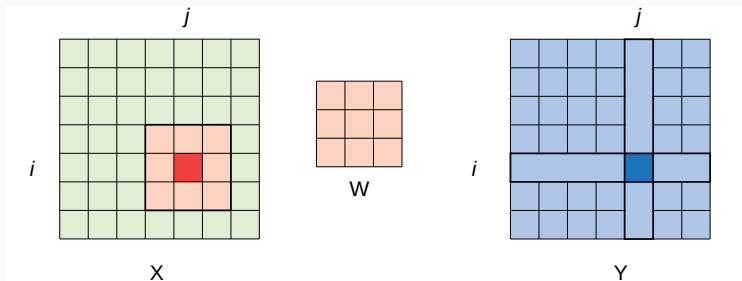
- The function h is the **kernel** representing L .
- In practice we use kernels with small support (e.g.: 3×3 pixels).
- This is **the basis of linear filtering and in particular CNNs!**

Analogy with linear image processing

Linear convolution (Riesz representation theorem)

The operator L is *linear and translation-invariant* if and only if there exists $h \in \mathcal{F}$ such that for any $f \in \mathcal{F}$

$$\forall x \in \mathbb{Z}^2, Lf(x) = f * h(x) = \sum_{y \in \mathbb{Z}^2} f(y)h(x - y).$$



$$Y_{ij} = \sum_{-p \leq k, l \leq p} X_{i+k, j+l} \cdot W_{k, l}.$$

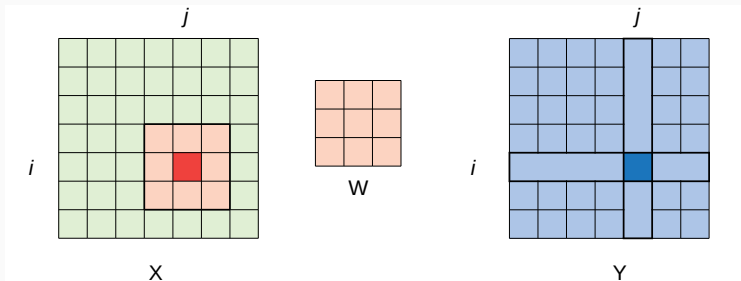
Analogy with linear image processing

Max-plus convolution

A dilation δ that is **translation invariant** and verifies $\delta(\lambda + f) = \lambda + \delta(f)$ for any image f and any constant λ , can be written as

$$\forall x \in \mathbb{Z}^2, \delta f(x) = \bigvee_{y \in \mathbb{Z}^2} f(y) + b(x - y)$$

where $b : \mathbb{Z}^2 \rightarrow \bar{\mathbb{R}}$ is called **structuring function** or **structuring element**.



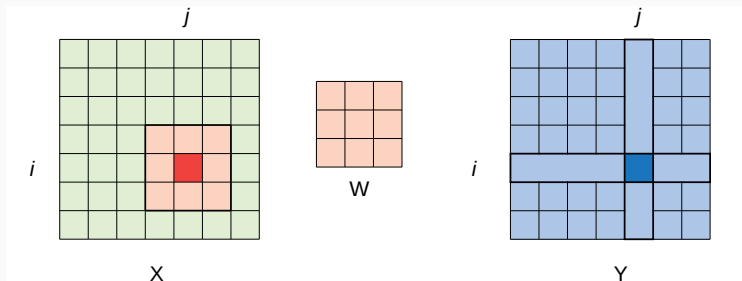
$$Y_{ij} = \bigvee_{-p \leq k, l \leq p} X_{i+k, j+l} + W_{k,l}$$

Analogy with linear image processing

Min-plus convolution

Similarly, a translation invariant erosion ε such that $\varepsilon(\lambda + f) = \lambda + \varepsilon(f)$, can be written as

$$\forall x \in \mathbb{Z}^2, \varepsilon f(x) = \bigwedge_{y \in \mathbb{Z}^2} f(y) + b'(x - y).$$



$$Y_{ij} = \bigwedge_{-p \leq k, l \leq p} X_{i+k, j+l} + W_{k, l}.$$

Vector spaces

Any translation invariant linear operator is a **linear convolution** with a **kernel** h
(Riesz representation theorem)

$$Lf(x) = f * h(x) = \sum_{y \in \mathbb{Z}^2} f(y)h(x - y).$$

Complete lattices

A translation invariant dilation is a **max-plus convolution** with a **structuring element** b

$$\delta f(x) = \bigvee_{y \in \mathbb{Z}^2} f(y) + b(x - y)$$

A translation invariant erosion is a **min-plus convolution** with a **structuring element** b

$$\varepsilon f(x) = \bigwedge_{y \in \mathbb{Z}^2} f(y) + b(x - y)$$

Morphological operators: dilations and erosions

Let $(\mathcal{L}, \vee, \wedge)$ and $(\mathcal{L}', \vee', \wedge')$ be two complete lattices.

Dilation

$\delta : \mathcal{L} \rightarrow \mathcal{L}'$ is a dilation if and only if for any family $(x_i)_{i \in I} \subseteq \mathcal{L}$,

$$\delta(\bigvee_i x_i) = \bigvee'_i \delta(x_i).$$

Erosion

$\varepsilon : \mathcal{L} \rightarrow \mathcal{L}'$ is an erosion if and only if for any family $(x_i)_{i \in I} \subseteq \mathcal{L}$,

$$\varepsilon(\bigwedge_i x_i) = \bigwedge'_i \varepsilon(x_i).$$

Morphological operators: openings and closings

From dilations and erosions, many morphological operators can be defined, in particular openings and closings.

Opening

An opening $\gamma : \mathcal{L} \rightarrow \mathcal{L}$ is an operator which is

- Increasing: $f \leq g \Rightarrow \gamma f \leq \gamma g$
- anti-extensive: $\gamma \leq id$
- idempotent: $\gamma \circ \gamma = \gamma$.

Closing

A closing $\varphi : \mathcal{L} \rightarrow \mathcal{L}$ is an operator which is

- Increasing: $f \leq g \Rightarrow \varphi f \leq \varphi g$
- extensive: $id \leq \varphi$
- idempotent: $\varphi \circ \varphi = \varphi$.

Adjunction

An erosion $\varepsilon : \mathcal{L} \rightarrow \mathcal{L}'$ and a dilation $\delta : \mathcal{L}' \rightarrow \mathcal{L}$ form an adjunction if and only if for any $x \in \mathcal{L}'$ and $y \in \mathcal{L}$

$$\delta(x) \leq y \iff x \leq \varepsilon(y).$$

Properties:

- Any dilation $\delta : \mathcal{L}' \rightarrow \mathcal{L}$ has a unique adjoint erosion $\varepsilon : \mathcal{L} \rightarrow \mathcal{L}'$ defined by

$$\forall y \in \mathcal{L}, \varepsilon(y) = \bigvee \{x \in \mathcal{L}', \delta(x) \leq y\}$$

- Any erosion $\varepsilon : \mathcal{L} \rightarrow \mathcal{L}'$ has a unique adjoint dilation $\delta : \mathcal{L}' \rightarrow \mathcal{L}$ defined by

$$\forall x \in \mathcal{L}', \delta(x) = \bigwedge \{y \in \mathcal{L}, x \leq \varepsilon(y)\}$$

- If (ε, δ) is an adjunction, then $\delta\varepsilon$ is an opening and $\varepsilon\delta$ is a closing.

Morphological operators: examples on images

We consider images defined on $E \subset \mathbb{Z}^2$ with integer values in $[0, 255]$. Then for any structuring function $b : \mathbb{Z}^2 \rightarrow [-255, 0] \cap \mathbb{N}$ such that for any $x \in E$, $\bigvee_{y \in E} b(y - x) = 0$, the operators δ_b and ε_b defined by

$$\forall x \in E, \delta_b f(x) = \bigvee_{y \in E} f(y) + b(y - x)$$

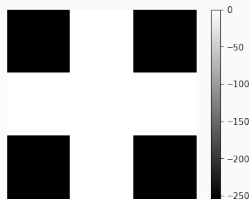
and

$$\forall x \in E, \varepsilon_b f(x) = \bigwedge_{y \in E} f(y) - b(x - y)$$

are adjoint dilation and erosion.

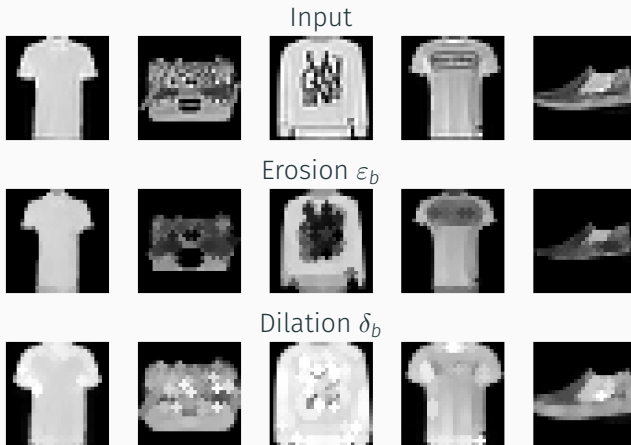
Morphological operators: examples on images

Cross flat structuring element:



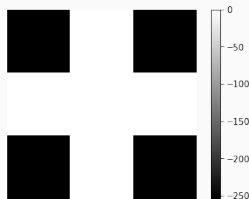
$$b(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ -255 & \text{otherwise} \end{cases}$$

$$\mathcal{C} = \{(0,0)\} \cup \left\{ \left(\cos\left(\frac{k\pi}{2}\right), \sin\left(\frac{k\pi}{2}\right) \right), 0 \leq k \leq 3 \right\}.$$



Morphological operators: examples on images

Cross flat structuring element:



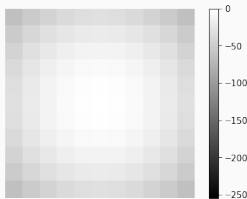
$$b(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ -255 & \text{otherwise} \end{cases}$$

$$\mathcal{C} = \{(0,0)\} \cup \left\{ \left(\cos\left(\frac{k\pi}{2}\right), \sin\left(\frac{k\pi}{2}\right) \right), 0 \leq k \leq 3 \right\}.$$



Morphological operators: examples on images

Quadratic structuring element:



$$b(x) = -c \cdot \frac{\|x\|^2}{t^2} \text{ with } c \geq 0.$$

Input



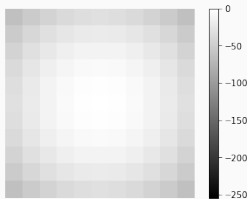
Erosion ε_b



Dilation δ_b

Morphological operators: examples on images

Quadratic structuring element:



$$b(x) = -c \cdot \frac{\|x\|^2}{t^2} \text{ with } c \geq 0.$$

Input



Opening $\gamma_b = \delta_b \varepsilon_b$

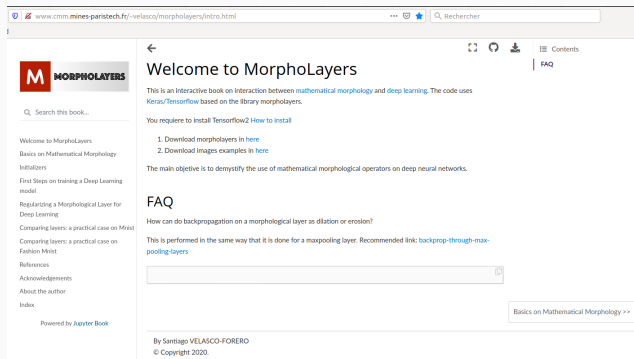


Closing $\varphi_b = \varepsilon_b \delta_b$

Learning structuring elements

Just like the kernels of translation invariant linear filters are learnt in convolutional neural networks (CNNs), structuring elements can be learnt in morphological neural networks!

This is the goal of **Morpholayers** *bit.ly/morpholayers*.



The screenshot shows the homepage of the Morpholayers project. The browser address bar displays `www.cmm.mines-paristech.fr/~velasco/morpholayers/intro.html`. The page features a left sidebar with a search bar and a list of navigation links: Welcome to MorphoLayers, Basics on Mathematical Morphology, Initializers, First Steps on training a Deep Learning model, Regularizing a Morphological Layer for Deep Learning, Comparing layers: a practical case on Mnist, Comparing layers: a practical case on Fashion Mnist, References, Acknowledgements, About the author, and Index. The main content area is titled "Welcome to MorphoLayers" and includes a description of the book as an interactive resource on mathematical morphology and deep learning, powered by Keras/Tensorflow. It lists requirements (Tensorflow2) and provides links for downloading the code and image examples. A FAQ section is also visible, with a question about backpropagation on morphological layers. The footer identifies the author as Santiago VELASCO-FORERO and notes the copyright year as 2020.

www.cmm.mines-paristech.fr/~velasco/morpholayers/intro.html

Rechercher

M MORPHOLAYERS

Search this book...

Welcome to MorphoLayers
Basics on Mathematical Morphology
Initializers
First Steps on training a Deep Learning model
Regularizing a Morphological Layer for Deep Learning
Comparing layers: a practical case on Mnist
Comparing layers: a practical case on Fashion Mnist
References
Acknowledgements
About the author
Index

Powered by Jupyter Book

Welcome to MorphoLayers

This is an interactive book on interaction between [mathematical morphology](#) and [deep learning](#). The code uses [Keras/Tensorflow](#) based on the library [morpholayers](#).

You require to install Tensorflow2 [How to install](#)

1. Download morpholayers in [here](#)
2. Download images examples in [here](#)

The main objective is to demystify the use of mathematical morphological operators on deep neural networks.

FAQ

How can do backpropagation on a morphological layer as dilation or erosion?

This is performed in the same way that it is done for a maxpooling layer. Recommended link: [backprop-through-max-pooling-layers](#)

Basics on Mathematical Morphology >>

By Santiago VELASCO-FORERO
© Copyright 2020.

Tutorial 1: Simple morphological operators using morpholayers.

<https://github.com/Jacobiano/morpholayers>

Mathematical morphology - Learning simple translation invariant operators

Learning structuring elements

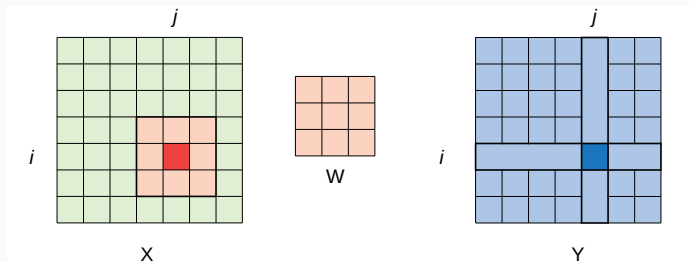
Learning structuring elements: a recurrent problem

Morphologists have been interested in **learning structuring elements** for a long time.

Yet, the problem is **not solved**!

See [Davidson and Hummer, 1993, Masci et al., 2013, Mondal et al., 2020, Kirszenberg et al., 2021].

Learning a dilation structuring element



Dilation layer

$$Y_{ij} = \bigvee_{-p \leq k, l \leq p} X_{i+k, j+l} + W_{k,l}.$$

Learning problem

- Fixed image $X \in \mathbb{R}^{m \times n}$
- Structuring element $W \in \mathbb{R}^{2p+1 \times 2p+1}$, to learn
- \tilde{X} : reshaped version of X , $m \times n$ lines, $(2p+1)^2$ columns. Each line is the $2p+1 \times 2p+1$ neighbourhood of one pixel.

The dilation δ with structuring element W is seen as the mapping

$$\delta_W : X \mapsto \tilde{X} \boxplus W$$

defined by

$$(\tilde{X} \boxplus W)_i = \bigvee_{1 \leq l \leq (2p+1)^2} \tilde{X}_{i,l} + W_l, \quad 1 \leq i \leq m \times n$$

Learning problem:

Given the input image X and an output image Y , find W such that $\delta_W(X) = Y$ (or $\approx Y$).

Morphological solution by adjunction

Let (ε, δ) be an adjunction, y a fixed element and consider the equation

$$(E) : \quad \delta(x) = y.$$

We know that

if (E) has a solution, then $x = \varepsilon(y)$ is a solution and it is the largest one.

Indeed, if $y = \delta(z)$ for some z , then $x = \varepsilon(y) = \varepsilon\delta(z)$ and therefore $\delta(x) = \delta\varepsilon\delta(z) = \delta(z) = y$.

Besides, $y = \delta(z)$ implies $\delta(z) \leq y$ and therefore $z \leq \varepsilon(y) = x$.

if (E) has no solution, then $x = \varepsilon(y)$ is the best under-approximation of a solution:

$$x = \varepsilon(y) = \bigvee \{z, \delta(z) \leq y\}.$$

Morphological solution by adjunction

In our case X and Y are known and we want to solve $\delta_W(X) = Y$ with respect to W .

But $\tilde{X} \boxplus W$ is also a dilation of W , which we may note it $\delta_{\tilde{X}}(W)$.

Its adjoint erosion $\varepsilon_{\tilde{X}} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{(2p+1)^2}$ is defined by

$$\varepsilon_{\tilde{X}}(Y)_l = (\tilde{X}^* \boxminus Y)_l = \bigwedge_{1 \leq i \leq m \times n} Y_i - \tilde{X}_{i,l}, \quad 1 \leq l \leq (2p+1)^2,$$

where $\tilde{X}^* = -\tilde{X}^T$.

Therefore if Y is the dilation of X by some structuring element, $W = \varepsilon_{\tilde{X}}(Y)$ is such a structuring element. If not, it is the best under-approximation of a solution.

Least square error solution with gradient descent

The learning problem may be formulated as minimizing a distance, e. g. the square Euclidean distance:

$$\min_W ||Y - \delta_W(X)||^2.$$

1. Is this a convex problem?
2. Can we solve it with gradient descent?

Least square error solution with gradient descent

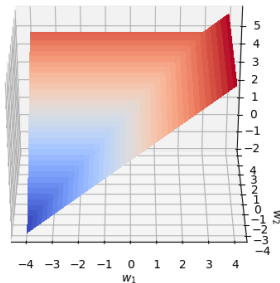
Convexity:

$$\|Y - \delta_W(X)\|^2 = \sum_{i=1}^{m \times n} (f_i(W) - Y_i)^2$$

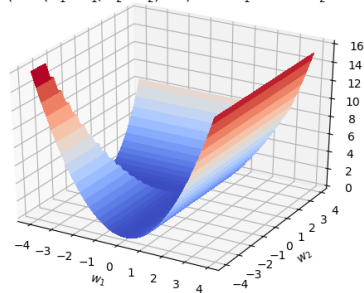
with $f_i(W) = (\tilde{X} \boxtimes W)_i = \bigvee_{1 \leq l \leq (2p+1)^2} \tilde{X}_{i,l} + W_l$.

Least square error solution with gradient descent

$$z = \max(w_1 + x_1, w_2 + x_2) \text{ with } x_1 = 2 \text{ and } x_2 = 1.$$



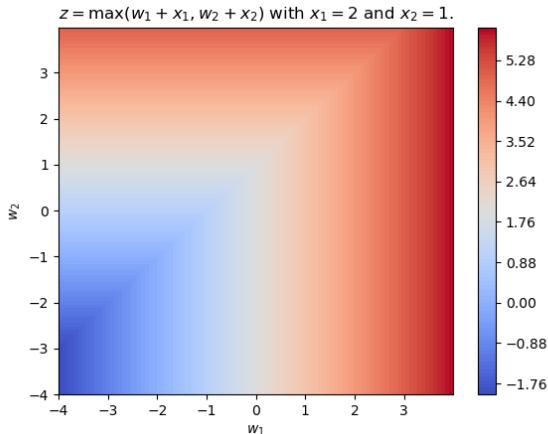
$$z = (\max(w_1 + x_1, w_2 + x_2) - 2)^2 \text{ with } x_1 = 2 \text{ and } x_2 = 1.$$



Left: An example of $f_i(W)$ in 2D. **Right:** $(f_i(W) - Y_i)^2$ for $Y_i = 2$; it is **not a convex function** of W (but it looks friendly anyway).

Least square error solution with gradient descent

Gradient:



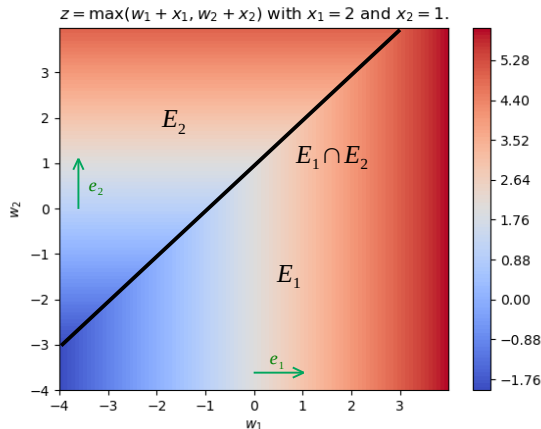
The function

$$\delta : \mathbf{w} \in \mathbb{R}^n \mapsto \bigvee_{1 \leq i \leq n} \{x_i + w_i\}$$

is differentiable almost everywhere
with respect to \mathbf{w} .

Least square error solution with gradient descent

Gradient:



Letting

$$E_j = \left\{ w \in \mathbb{R}^n, x_j + w_j = \bigvee_{1 \leq i \leq n} \{x_i + w_i\} \right\},$$

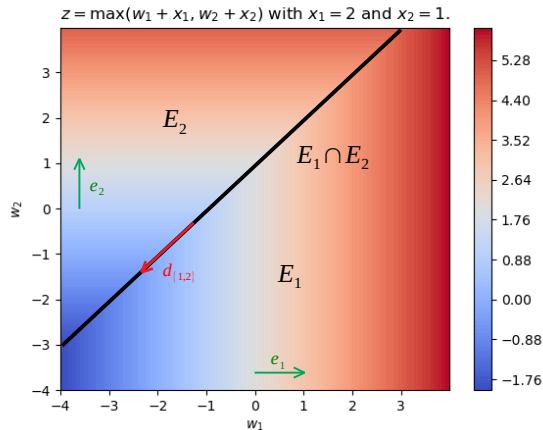
then δ is differentiable on the interior of E_j and

$$\forall w \in \overset{\circ}{E}_j, \quad \nabla_w \delta(w, x) = e_j$$

where e_j is the j th canonical base vector of \mathbb{R}^n ($e_{ij} = 1$ if $i = j$ and 0 otherwise).

Least square error solution with gradient descent

Gradient:



Besides, let $I \subseteq \{1, \dots, n\}$. Then on $\bigcap_{i \in I} E_i \setminus (\bigcup_{j \notin I} E_j)$

$$d_I = - \sum_{i \in I} e_i$$

is a descent direction of δ .

Least square error solution with gradient descent

Gradient:

For our problem, noting $\ell(W) = \|Y - \delta_W(X)\|^2$, we have

$$\nabla \ell(W) = 2G^T \cdot [\tilde{X} \boxtimes W - Y]$$

where $G \in \{0, 1\}^{m \cdot n \times (2p+1)^2}$, and

$$G_{i,k} = \begin{cases} 1 & \text{if } \tilde{X}_{ik} + W_k = \bigvee_{1 \leq l \leq (2p+1)^2} \tilde{X}_{i,l} + W_l \\ 0 & \text{otherwise.} \end{cases}$$

Learning an opening structuring element

The opening γ with structuring element W is seen as the mapping

$$\gamma_W : X \mapsto \delta_W(\varepsilon_W(X))$$

with

$$\varepsilon_W(X)_i = (\tilde{X} \boxminus W^*)_i = \bigwedge_{1 \leq l \leq (2p+1)^2} \tilde{X}_{i,l} - W_l^T, \quad 1 \leq i \leq m \times n$$

and as before

$$\delta_W(Y)_i = (\tilde{Y} \boxplus W)_i = \bigvee_{1 \leq l \leq (2p+1)^2} \tilde{Y}_{i,l} + W_l, \quad 1 \leq i \leq m \times n$$

(Here W^T denotes column vector, like W . W is the reshaped version of a matrix A , and W^T of A^T).

Learning problem:

Given the input image X and an output image Z , find W such that $\gamma_W(X) = Z$ (or $\approx Z$).

Learning an opening structuring element

Learning problem:

Given the input image X and an output image Z , find W such that $\gamma_W(X) = Z$ (or $\approx Z$).

The problem can also be formulated with an auxiliary variable Y :

$$\begin{cases} \tilde{Y} \boxtimes W &= Z \\ \tilde{X} \boxtimes W^* &= Y \end{cases}$$

As both Y and W are unknown, the first equation is actually a **matrix factorization problem in the max-plus algebra** (see [Karaev and Miettinen, 2019] for algorithms).

The second equation can be seen as a criterion to choose among several approximate solutions of the first line.

Matrix factorization is a **difficult problem**. Why not try gradient descent?

Learning an opening structuring element with gradient descent

Two alternative architectures:

- The **one layer** architecture: an opening layer parameterized by one structuring element W

$$\gamma_W(X) = \delta_W(\varepsilon_W(X))$$

- The **two layers** architecture: an erosion layer followed by a dilation layer with independent parameters W_1 and W_2

$$g_{\underline{W}}(X) = \delta_{W_2}(\varepsilon_{W_1}(X))$$

where $\underline{W} = [W_1, W_2]$.

Learning an opening structuring element with gradient descent

Two alternative gradients!

Noting:

- ∂_w : derivative with respect to the parameter
- ∂_x : derivative with respect to the input
- ℓ : loss function, typically $\ell(\hat{Y}) = \|\hat{Y} - Y\|_2^2$ for a fixed Y (label associated to X)
- $\Gamma(W, X) = \ell(\gamma_W(X)) \in \mathbb{R}$
- $G(\underline{W}, X) = \ell(g_{\underline{W}}(X)) \in \mathbb{R}$

we get

$$\nabla_w \Gamma(W, X) = \nabla \ell(\gamma_W(X)) \cdot \left[\partial_w \delta_W(\varepsilon_W(X)) + \partial_x \delta_W(\varepsilon_W(X)) \cdot \partial_w \varepsilon_W(X) \right]^T$$

$$\nabla_w G(\underline{W}, X) = \nabla \ell(g(\underline{W}, X)) \cdot \left[\begin{array}{c} \partial_w \delta_{W_2}(\varepsilon_{W_1}(X)) \\ \partial_x \delta_{W_2}(\varepsilon_{W_1}(X)) \cdot \partial_w \varepsilon_{W_1}(X) \end{array} \right]^T$$

Tutorial 2: Learning morphological operators.

<https://github.com/Jacobiano/morpholayers>

Conclusions on learning translation invariant morphological operators

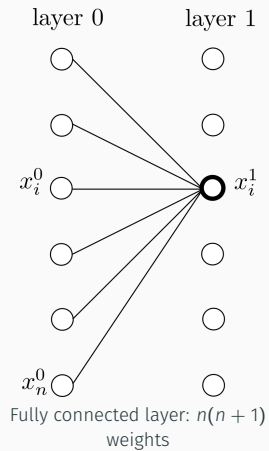
- Training neural networks including **morphological layers** is a **natural** counterpart of classical CNN training
- **Tools already exist** to experiment and discover new insights (we presented **Tensorflow + Morpholayers**)
- The field is also related to algebraic problems in **tropical algebra** (equation solving, matrix factorization...); studying these problems could lead to alternative training strategies.

Note: Morphological neural networks can achieve **much more than learning structuring elements!**

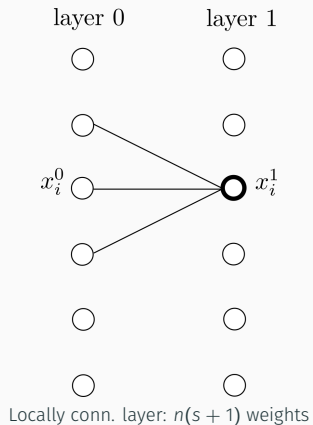
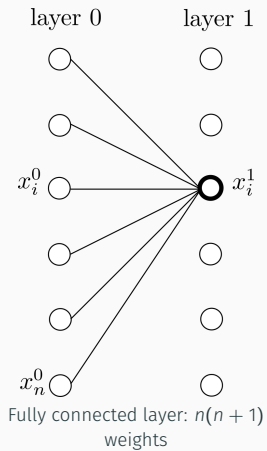
See [Charisopoulos and Maragos, 2017, Zhang et al., 2019, Blusseau et al., 2020, Franchi et al., 2020, Maragos et al., 2021, Tsilivis et al., 2021].

Depthwise Morphological Layers

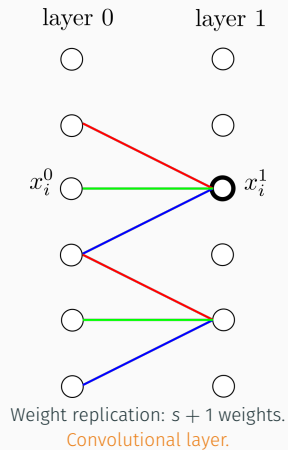
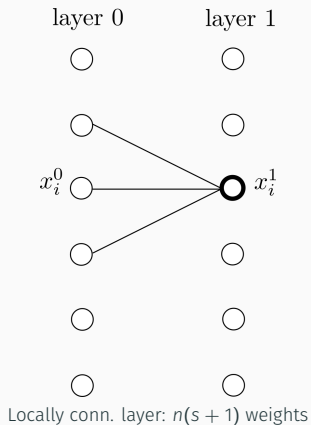
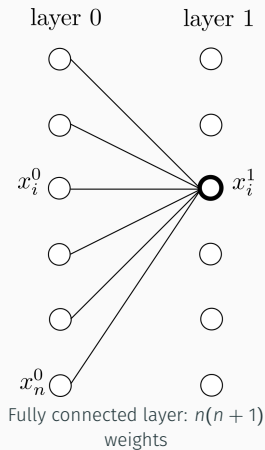
Towards convolutional layers



Towards convolutional layers



Towards convolutional layers



Convolutional Filter

Recall that strictly speaking, convolutional layers are a *misnomer*, since the operations they express are more accurately described as cross-correlations.

Given an input image $\mathbf{I} \in \mathbb{R}^{M \times N}$ and a filter (kernel) \mathbf{K} of dimensions $k_1 \times k_2$, the **cross-correlation** operation is given by:

$$(\mathbf{I} * \mathbf{K})_{ij} := \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \mathbf{I}(i+m, j+n) \mathbf{K}(m, n), \quad (3)$$

$\forall i \in 1, \dots, M$ and $j \in 1, \dots, N$ Note that $(\mathbf{I} * \mathbf{K})_{ij}$ is a mapping $\mathbb{R}^{M \times N} \mapsto \mathbb{R}^{M \times N}$

Classical Convolution is the same as cross-correlation with a flipped kernel. Anyways we will use the term "convolutional layers".

Zero-Padding: In this talk, we consider zero-padding in the computation of convolution. However in implementation this should be specified.

Example of Architecture

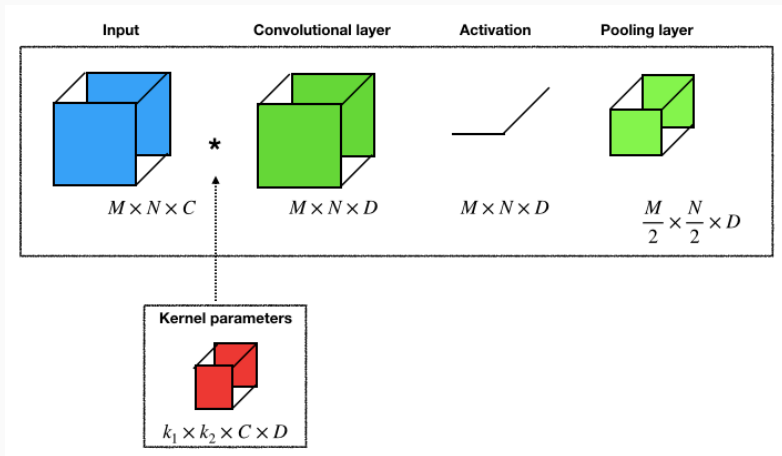


Figure 2: A level of typical architecture used in DL

Tutorial 3: Learning morphological layers in Fashion Mnist
<https://github.com/Jacobiano/morpholayers>

Multivariate Convolutional Filter

CNNs consists of convolutional layers which are characterized by an input map \mathbf{I} , a bank of filters \mathbf{K} and biases b .

In the case of images, we could have as input an image with height H , width W and $C = 3$ channels (red, blue and green) such that $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$. Subsequently for a bank of D filters we have $\mathbf{K} \in \mathbb{R}^{k_1 \times k_2 \times C \times D}$ and biases $\mathbf{b} \in \mathbb{R}^D$, one for each filter.

The output from this convolution procedure is as follows:

$$[(\mathbf{I} * \mathbf{K})_{ij}]_d := \sum_{c=1}^C \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \mathbf{I}(i+m, j+n, c) \mathbf{K}(m, n, c, d) + \mathbf{b}(d) \quad (4)$$

$\forall i \in 1, \dots, M$ and $j \in 1, \dots, N$. Note that $[(\mathbf{I} * \mathbf{K})_{ij}]_d$ is a mapping $\mathbb{R}^{M \times N \times C} \mapsto \mathbb{R}^{M \times N \times D}$

"Classical convolutional filters" in Multi-valued images

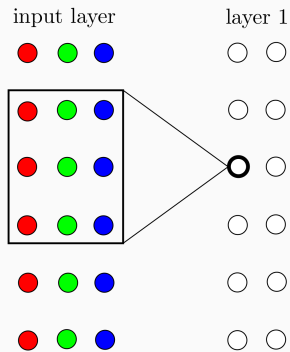
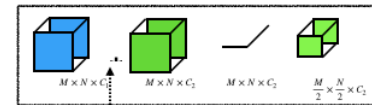
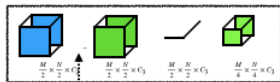


Figure 3: Note that operators combine multivalued information.

Example of Architecture



Convolution Activation Pooling



Convolution Activation Pooling



Flatten + MLP

$$((\frac{M}{4} \times \frac{N}{4} \times C_3) + 1) \times \text{classes}$$

Downsampling operators

- **Convolutions with strides:** We move our window more than one element at a time, skipping the intermediate locations. We refer to the number of rows and columns traversed per slide as the **stride**.

The operator convolution with strides (s_1, s_2) , is defined as

$$[(I * K)_{ij}]_d, \forall i \in s_1 - 1, 2s_1 - 1, 3s_1 - 1, \dots, M \text{ and } \forall j \in s_2 - 1, 2s_2 - 1, 3s_2 - 1, \dots, N.$$

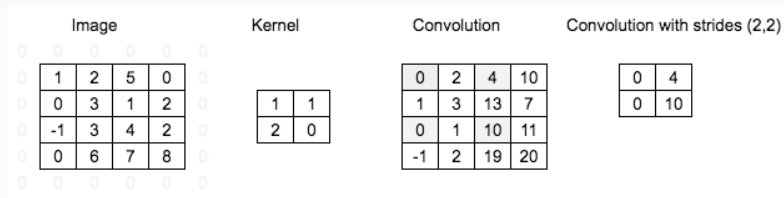


Figure 5: Example of a convolution with strides (2,2)

$[(I * K)_{ij}]_d$ with strides (s_1, s_2) is a mapping $\mathbb{R}^{M \times N \times C} \mapsto \mathbb{R}^{M/s_1 \times N/s_2 \times C}$

Downsampling operators (Pooling Layers)

- **Pooling Layers** The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Like convolutional layers, pooling operators consist of a fixed-shape window that is slid over all regions in the input according to its stride, computing a single output for each location traversed by the fixed-shape window (sometimes known as the pooling window)

$$\text{MaxPooling}[I_{ij}]_c := \bigvee_{m=0}^{k_1-1} \bigvee_{n=0}^{k_2-1} I(i+m, j+n, c) \quad (5)$$

$$\text{AveragePooling}[I_{ij}]_c := \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n, c) / (k_1 * k_2) \quad (6)$$

$\forall i \in s_1 - 1, 2s_1 - 1, 3s_1 - 1, \dots, M$ and $\forall j \in s_2 - 1, 2s_2 - 1, 3s_2 - 1, \dots, N$

Downsampling operators (Pooling Layers)

- **Pooling Layers** The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Like convolutional layers, pooling operators consist of a fixed-shape window that is slid over all regions in the input according to its stride, computing a single output for each location traversed by the fixed-shape window (sometimes known as the pooling window)

$$\text{MaxPooling}[I_{ij}]_c := \bigvee_{m=0}^{k_1-1} \bigvee_{n=0}^{k_2-1} I(i+m, j+n, c) \quad (5)$$

$$\text{AveragePooling}[I_{ij}]_c := \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n, c) / (k_1 * k_2) \quad (6)$$

$\forall i \in s_1 - 1, 2s_1 - 1, 3s_1 - 1, \dots, M$ and $\forall j \in s_2 - 1, 2s_2 - 1, 3s_2 - 1, \dots, N$

Example Pooling

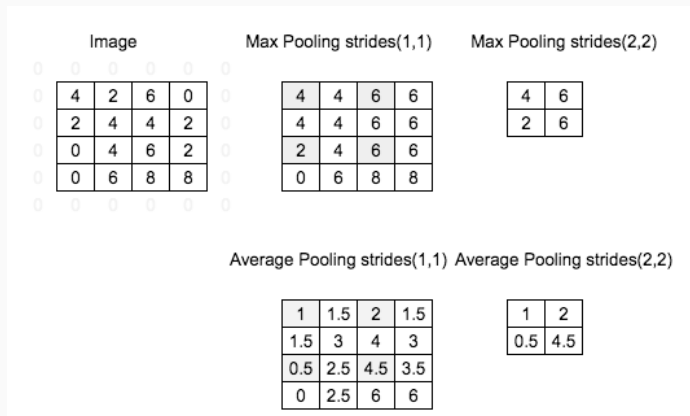


Figure 6: Example of Average and Max-Pooling with strides (2,2)

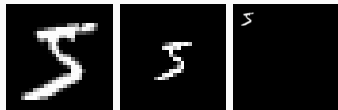
- Note that $\text{Pooling}[(I * K)_{ij}]_d$ with strides (s_1, s_2) is a mapping $\mathbb{R}^{M \times N \times C} \mapsto \mathbb{R}^{M/s_1 \times N/s_2 \times C}$
- Note that Max-Pooling is a Flat Dilation with strides applied channel by channel (in

Tutorial 4: Improving Max-Pooling layers using Dilations
<https://github.com/Jacobiano/morpholayers>

Morphological Scale-Spaces: Introduction to Invariance and Equivariance

Morphological Scale-Spaces: Introduction to Invariance and Equivariance

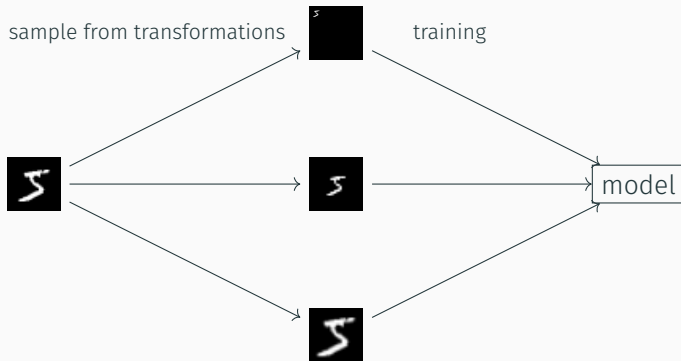
Equivariance and Invariance



- A property that is preserved under certain transformations on the data
- As an example, all of the images above represent the digit “five” although they differ by scale and translation
- Formally, an operator ψ is **invariant** to a set of transformations $\{L_g | g \in G\}$ if $\forall g \in G$

$$\psi \circ L_g = \psi$$

Data Augmentation



- No theoretical assurance of invariance
- Needs more parameters

Group and Semigroup Actions

Definition

A tuple (G, \cdot) is a **group** if

1. $\forall g, h, k \in G, (g \cdot h) \cdot k = g \cdot (h \cdot k)$;
2. $\exists e \in G, \forall g \in G, e \cdot g = g \cdot e = g$;
3. $\forall g \in G, \exists g^{-1}, g^{-1} \cdot g = g \cdot g^{-1} = e$.

Examples:

- Rotation:



- Scaling:



- Occlusion(semigroup):



- Translation:



- Scaling and translation:

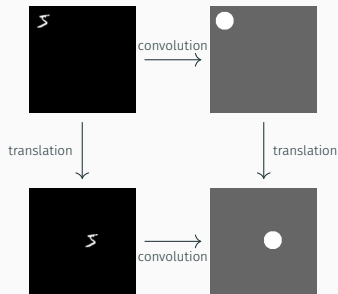


- Shear:



- A tuple (G, \cdot) that satisfies the first two properties is called a **semigroup**.
- A group(semigroup) **action** is a family of operators $L_g : X \rightarrow X, g \in G$ that satisfies $L_g \circ L_h = L_{g \cdot h}$ or $L_g \circ L_h = L_{h \cdot g}$

Equivariance



Let $l : \mathbb{Z}^d \rightarrow \mathbb{R}$, $h : \mathbb{Z}^d \rightarrow \mathbb{R}$ their convolution is given by

$$(l \star h)(x) = \sum_{y \in \mathbb{Z}^d} l(y)h(x - y) = \sum_{y \in \mathbb{Z}^d} R_x(l)(y)h(-y)$$

where $R_x(l)(y) = l(x + y)$. We have $R_z(l) \star h = R_z(l \star h)$.

Equivariance Formal Definition

Given an operator $\psi : X \rightarrow Y$ and a (semi)group G , then we say that ψ is **equivariant** with respect to G if there exists two actions $L_g, L'_g, g \in G$, on X and Y , respectively, such that, for all $I \in X$

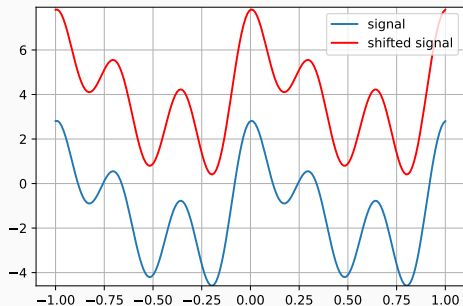
$$\psi[L_g(I)] = L'_g[\psi(I)]$$

- In the previous example, we saw that convolution is **group equivariant** with respect to translations.
- The same is true for morphological dilations, erosions, openings and closings.
- We will also investigate two important equivarances of some morphological operators, namely the equivariance to additive shift and to scalings.

Equivariance to Additive Shift

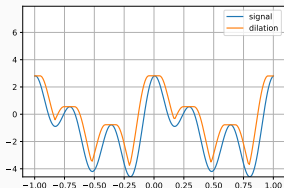
Morphological dilations, erosions, openings and closings are equivariant to **additive shifts**, i.e. the operation of adding a constant to the output of a signal or image $(I + t)(x) = I(x) + t$, globally changing its brightness

- $\delta(I + t) = \delta(I) + t$
- $\varepsilon(I + t) = \varepsilon(I) + t$
- $\gamma(I + t) = \gamma(I) + t$
- $\varphi(I + t) = \varphi(I) + t$

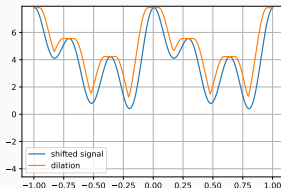


Invariance to Additive Shift

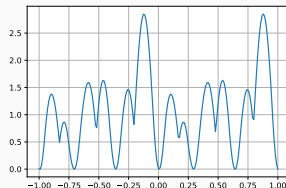
Morphological gradients and top-hats are invariant to additive shift



(a) f and $\delta(f)$



(b) $f + t$ and $\delta(f + t) = \delta(f) + t$



(c) $\delta(f + t) - (f + t) = \delta(f) - f$

Invariance to Additive Shift



(a) I



(b) $\delta(I) - I$



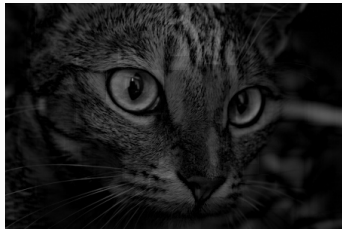
(c) $I - \gamma(I)$



(d) $I + t$



(e) $\delta(I + t) - (I + t) = \delta(I) - I$



(f) $I + t - \gamma(I + t) = I - \gamma(I)$

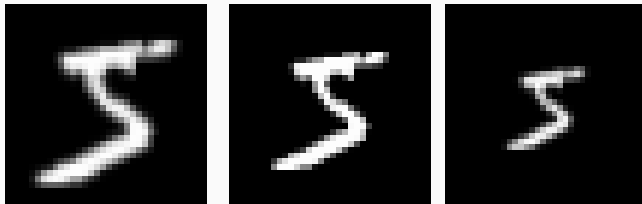
Tutorial 5: Learning Additive Shift Invariant Operators
<https://github.com/Jacobiano/morpholayers>

Morphological Scale-Spaces: Introduction to Invariance and Equivariance

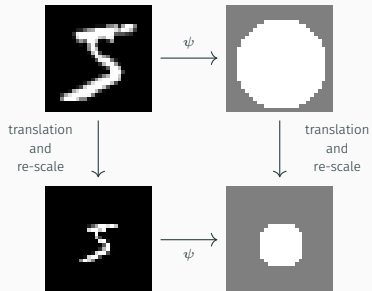
Morphological Scale-Spaces

Re-Scaling

- The action of **scaling** can be written as $(R'_s f)(x) = f(s^{-1}x)$
- When scaling is viewed as a group, we can have either zoom in ($s > 1$) or zoom out ($s < 1$)
- When it is modeled as a semigroup, we consider on the case of zoom out
- Here, we model it as a semigroup case in order to model discrete scalings, in this way, no information is created through interpolation

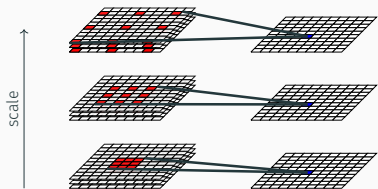


Scale Equivariance



Scale-Crosscorrelation [Worrall and Welling, 2019]

$$(l \star_G h)(2^{-k}, x) = \sum_{l \geq 0} \sum_{y \in \mathbb{Z}^2} l(2^{-k-l}, 2^k y + x) h(2^{-l}, y),$$



- The scale-crosscorrelation is equivariant with respect to the semigroup of scalings
- It assumes that the input domain of l is $G = \mathcal{S} \times \mathbb{Z}^2$, a semigroup of scales and translations
- We use (morphological) **scale-spaces** to map l to the adequate space

Scale-Spaces

- Here we consider as scale spaces, operators that commute with a scaling operator,

$$T_t[R'_s(I)] = R'_s[T_{\frac{t}{s^p}}(I)]$$

for some integer p . This implies that they are scale-equivariant.

- An example is the Gaussian Scale-Space: $T_{\mathcal{G}_t}(I) = I \star \mathcal{G}_t$, where

$$\mathcal{G}_t(x) = (4\pi t)^{-1} \exp\left(-\frac{\|x\|^2}{4t}\right), t > 0$$



- We have $T_{\mathcal{G}_t} \circ R'_s = R'_s \circ T_{\mathcal{G}_{t/s^2}}$

- Usually obtained from PDEs, e.g., the Gaussian scale-space $T_{G_t}[f](x, y) = u(t, x, y)$ is obtained from

$$\begin{cases} u_t = \operatorname{div}(\nabla u) = u_{xx} + u_{yy}, \\ u(0, x, y) = f(x, y) \end{cases}$$

- **Morphological** scale-spaces arise from the Hamilton-Jacobi equation $u_t = H(x, \nabla u)$
- **Quadratic** morphological scale-spaces come from

$$\begin{cases} u_t = \pm \|\nabla u\|^2, \\ u(0, x, y) = f(x, y) \end{cases}$$

where a positive sign yields a dilation scale-space and a negative sign yields an erosion scale-space.

Quadratic Dilations and Erosions

$$\delta_t(I)(x) = \bigvee_{y \in \mathbb{R}^d} I(x - y) - c \cdot \frac{\|y\|^2}{4t},$$

$$\varepsilon_t(I)(x) = \bigwedge_{y \in \mathbb{R}^d} I(x + y) + c \cdot \frac{\|y\|^2}{4t}.$$

The parameter c is optimized along with the weights of the network.

Quadratic Openings and Closings

$$\gamma_t = \delta_t \circ \varepsilon_t,$$

$$\varphi_t = \varepsilon_t \circ \delta_t.$$

Properties

- Commutation with scaling:

$$\delta_t \circ R'_s = R'_s \circ \delta_{t/s^2}$$

$$\varepsilon_t \circ R'_s = R'_s \circ \varepsilon_{t/s^2}$$

$$\gamma_t \circ R'_s = R'_s \circ \gamma_{t/s^2}$$

$$\varphi_t \circ R'_s = R'_s \circ \varphi_{t/s^2}$$

- Composition:

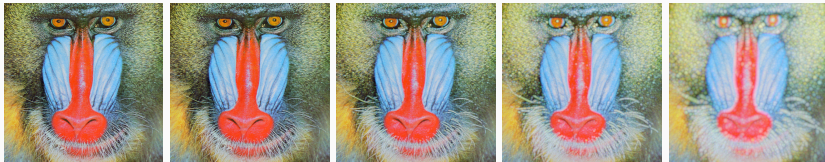
$$\delta_t \circ \delta_s = \delta_{t+s}$$

$$\varepsilon_t \circ \varepsilon_s = \varepsilon_{t+s}$$

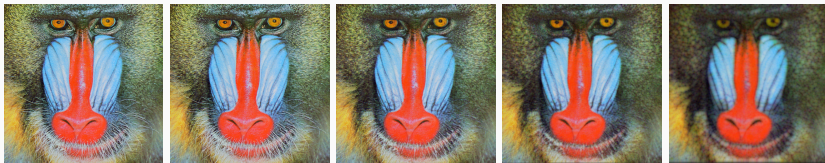
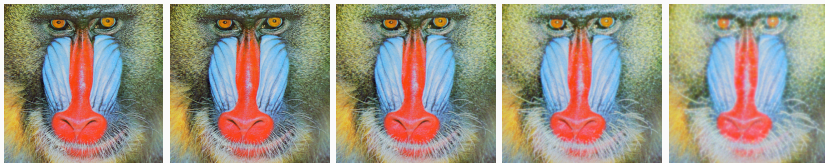
$$\gamma_t \circ \gamma_s = \gamma_{t \vee s}$$

$$\varphi_t \circ \varphi_s = \varphi_{t \vee s}$$

Quadratic Dilations and Erosions

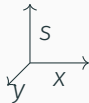
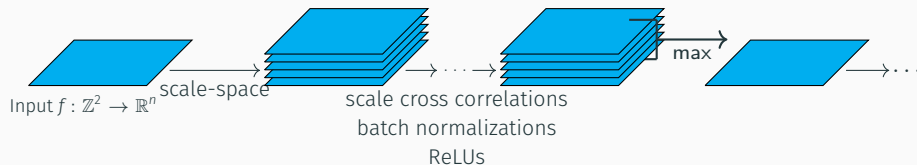


Quadratic Closings and Openings






Scale-Equivariant Networks




- Here we apply the morphological scale-spaces to create a scale-equivariant architecture using the scale-crosscorrelation [Sangalli et al., 2021, Worrall and Welling, 2019] operator
- The scale-crosscorrelation needs images defined on a space of scales
- We use the scale-spaces in order to **lift** images to those spaces









Tutorial 6: Learning Scale Equivariant Operators




<https://github.com/Jacobiano/morpholayers>

-  Blusseau, S., Ponchon, B., Velasco-Forero, S., Angulo, J., and Bloch, I. (2020).
Approximating morphological operators with part-based representations learned by asymmetric auto-encoders.
Mathematical Morphology-Theory and Applications, 4(1):64–86.
-  Cauchy, A. (1847).
Méthode générale pour la résolution des systèmes d'équations simultanées.
arXiv preprint arXiv:1608.03983.
-  Charisopoulos, V. and Maragos, P. (2017).
Morphological perceptrons: geometry and training algorithms.
In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 3–15. Springer.

-  Davidson, J. L. and Hummer, F. (1993).
Morphology neural networks: An introduction with applications.
Circuits, Systems and Signal Processing, 12(2):177–210.
-  Franchi, G., Fehri, A., and Yao, A. (2020).
Deep morphological networks.
Pattern Recognition, 102:107246.
-  Karaev, S. and Miettinen, P. (2019).
Algorithms for approximate subtropical matrix factorization.
Data Mining and Knowledge Discovery, 33(2):526–576.

-  Kirszenberg, A., Tochon, G., Puybareau, É., and Angulo, J. (2021).
Going beyond p-convolutions to learn grayscale morphological operators.
In International Conference on Discrete Geometry and Mathematical Morphology, pages 470–482. Springer.
-  Maragos, P., Charisopoulos, V., and Theodosis, E. (2021).
Tropical geometry and machine learning.
Proceedings of the IEEE, 109(5):728–755.
-  Masci, J., Angulo, J., and Schmidhuber, J. (2013).
A learning framework for morphological operators using counter-harmonic mean.
In International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing, pages 329–340. Springer.

-  Mondal, R., Dey, M. S., and Chanda, B. (2020).
Image restoration by learning morphological opening-closing network.
Mathematical Morphology - Theory and Applications, 4(1):87–107.
-  Robbins, H. and Monro, S. (1985).
A stochastic approximation method.
In *Herbert Robbins Selected Papers*, pages 102–109. Springer.
-  Sangalli, M., Blusseau, S., Velasco-Forero, S., and Angulo, J. (2021).
Scale equivariant neural networks with morphological scale-spaces.
In *International Conference on Discrete Geometry and Mathematical Morphology*, pages 483–495. Springer.

-  Tsilivis, N., Tsiamis, A., and Maragos, P. (2021).
Sparse approximate solutions to max-plus equations.
In International Conference on Discrete Geometry and Mathematical Morphology, pages 538–550. Springer.
-  Worrall, D. E. and Welling, M. (2019).
Deep scale-spaces: Equivariance over scale.
arXiv preprint arXiv:1905.11697.
-  Zhang, Y., Blusseau, S., Velasco-Forero, S., Bloch, I., and Angulo, J. (2019).
Max-plus operators applied to filter selection and model pruning in neural networks.
In International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing, pages 310–322. Springer.