# 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, SaintÉtienne, 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.

PSL*

## 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
- Seven online tutorials.


# Deep Learning in 15 minutes 

## Learning from data

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


## Learning from data

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


Figure 1: Supervised classification

The pairs $\left(\mathrm{x}_{k}, \mathrm{y}_{k}\right)$ define the nature of the problem, supervised, denoising, segmentation, image transformation, and so on.

## Deep Neural Network

- A Deep Neural Network with $d$ layers and parameters $\boldsymbol{\theta}$, in its simplest form, is a compositional map that may be written as:

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

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, $f^{(i)}(\mathrm{x}):=g^{(i)}\left(\mathrm{W}^{(i)} \mathbf{x}+\mathrm{b}^{(i)}\right)$, with $g^{(i)}$ is a non-linear function (activation), and $\boldsymbol{\theta}=\left[\mathbf{W}^{(i)}, \mathbf{b}^{(i)}\right]$ (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 $\boldsymbol{\theta}$ so as to minimise the difference between the labels (ideal maps $\left\{\mathbf{y}_{k}\right\}$ ) and the outputs of the DNN (estimated maps $\left\{\hat{y}_{k}:=\phi_{f ; \theta}\left(\mathrm{x}_{k}\right)\right\}$ ) at any training instance $\mathrm{x}_{k}$, such that the difference goes to zero as the number of samples $n$ increases.
- For that, we define a loss function denoted by Loss : $\left.(\mathcal{Y} \times \mathcal{Y}) \mapsto \mathbb{R}^{+}\right)$, 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 $\left\{\left(\mathbf{x}_{k}, y_{k}\right)\right\}$

$$
\begin{equation*}
\hat{\boldsymbol{\theta}}:=\arg \min \sum_{i=1}^{n} \operatorname{Loss}\left(\mathrm{y}_{k}, \phi_{f ; \boldsymbol{\theta}}\left(\mathrm{x}_{k}\right)\right) \tag{1}
\end{equation*}
$$

## Minimization by backpropagration and SGD

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

1. SGD starts from certain initial $\boldsymbol{\theta}$ (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 $\nabla_{\boldsymbol{\theta}} \operatorname{Loss}(\mathrm{x})$. (Back-propagation)
4. Updates each parameter by moving it in the direction of the negative gradient, $\boldsymbol{\theta}_{\mathrm{t}+1}=\boldsymbol{\theta}_{\mathrm{t}}-\eta \nabla_{\boldsymbol{\theta}_{\mathrm{t}}} \operatorname{Loss}(\mathrm{x})$, where $\eta$ 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 $\boldsymbol{\theta}_{i}$ via the composition of several simple functions. In order to compute the gradient $\nabla_{\boldsymbol{\theta}} \operatorname{Loss}(\mathrm{x})$ we will make extensive use of the vector chain rule, i.e, $\frac{\partial}{\partial \mathrm{x}} \mathrm{f}(\mathrm{g}(\mathrm{x}))=\frac{\partial \mathrm{f}}{\partial \mathrm{g}} \frac{\partial \mathrm{g}}{\partial \mathrm{x}}$.

$$
\begin{equation*}
\nabla_{\boldsymbol{\theta}} \operatorname{Loss}\left(\mathrm{y}_{k}, \boldsymbol{\phi}_{f, \boldsymbol{\theta}}\left(\mathrm{x}_{k}\right)\right)=\frac{\partial \operatorname{Loss}}{\partial \phi_{f, \boldsymbol{\theta}}\left(\mathrm{x}_{k}\right)} \frac{\partial \boldsymbol{\phi}_{f, \boldsymbol{\theta}}(\mathrm{x})}{\partial \boldsymbol{\theta}} \tag{2}
\end{equation*}
$$

## Backpropagation applied to neural networks

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

$$
\begin{equation*}
\nabla_{\boldsymbol{\theta}} \operatorname{Loss}\left(\mathrm{y}_{k}, \boldsymbol{\phi}_{f, \boldsymbol{\theta}}\left(\mathrm{x}_{k}\right)\right)=\frac{\partial \operatorname{Loss}}{\partial \phi_{f, \boldsymbol{\theta}}\left(\mathrm{x}_{k}\right)} \frac{\partial \boldsymbol{\phi}_{f, \boldsymbol{\theta}}(\mathrm{x})}{\partial \boldsymbol{\theta}} \tag{2}
\end{equation*}
$$

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 X .

```
Algorithm 1 pseudocode gradient descent
    1: given initial learning rate }\eta\in\mathbb{R}\mathrm{ and dataset X
    initialize time step t=0, parameter vector 偖=0}\in\mp@subsup{\mathbb{R}}{}{P}\mathrm{ ,
    repeat
    4: t=t+1
    5: }\quad\mp@subsup{\textrm{g}}{t}{}=\nabla\mp@subsup{\operatorname{Loss}}{i}{}(\textrm{X},\mp@subsup{\boldsymbol{0}}{t-1}{})\mathrm{ Return gradient via backpropagation
    6:}\quad\mp@subsup{\boldsymbol{0}}{t}{}=\mp@subsup{\boldsymbol{0}}{t-1}{}-\eta\mp@subsup{g}{t}{
    7: until stopping criterion is met
    8: return optimized parameters }\mp@subsup{\boldsymbol{0}}{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 \(\mathbf{X}\)
    2: initialize time step \(t=0\), parameter vector \(\boldsymbol{\theta}_{t=0} \in \mathbb{R}^{P}\),
    3: repeat
    4: \(\mathrm{t}=\mathrm{t}+1\)
    5: \(\quad X_{t}=\operatorname{SelectBatch}(X)\) select a batch from data, whole data, only one, \(\ldots\)
    6: \(\quad g_{t}=\nabla\) Loss \(_{i}\left(X_{t}, \boldsymbol{\theta}_{t-1}\right)\) Return gradient via backpropagation
    7: \(\quad \boldsymbol{\theta}_{t}=\boldsymbol{\theta}_{t-1}-\eta g_{t}\)
    8: until stopping criterion is met
    : return optimized parameters \(\boldsymbol{\theta}_{i}\)
```

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 $\wedge A$.

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

- $\bigvee \mathcal{P}(E)=E$
- $\wedge \mathcal{P}(E)=\emptyset$
$A \subset E$



## 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 $\wedge A$.

Example 2: $\left([a, b]^{n}, \subseteq_{n}\right)$, with $a \leq b \in \mathbb{R}, n \in \mathbb{N}^{*}$ and $\forall x, y \in[a, b]^{n}, x \leq_{n} y \Longleftrightarrow x_{i} \leq y_{i} \forall i \in\{1, \ldots, n\}$

- $\mathrm{V}[a, b]^{n}=[b \ldots b]^{\top}=\mathrm{b}$
- $\wedge[a, b]^{n}=[a \ldots a]^{\top}=a$
- $\bigvee_{j \in J} x^{(j)}=\left[v_{j} x_{1}^{(j)}, \ldots, v_{j} x_{n}^{(j)}\right]$
- $\Lambda_{j \in J} x^{(j)}=\left[\wedge_{j} x_{1}^{(j)}, \ldots, \wedge_{j} x_{n}^{(j)}\right]$

Typically, 8-bits images:
$a=0, b=255$ and $n$ the number of pixels.


## Analogy with linear image processing

## Vector spaces

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

$$
f\left(\sum_{i} \lambda_{i} x_{i}\right)=\sum_{i} \lambda_{i} f\left(x_{i}\right)
$$

## Complete lattices

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

$$
\delta\left(\bigvee_{i} x_{i}\right)=\bigvee_{i} \delta\left(x_{i}\right)
$$

- Mappings commuting with the infimum are the erosions.

$$
\varepsilon\left(\bigwedge x_{i}\right)=\bigwedge \varepsilon\left(x_{i}\right)
$$

## 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 $u \in \mathbb{Z}^{2}, L \circ \tau_{u}=\tau_{u} \circ L$.


## 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 $f \in \mathcal{F}$

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

- The function $h$ is the kernel representing $L$.
- In practice we use kernels with small support (e.g.: $3 \times 3$ 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 $f \in \mathcal{F}$

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



## Analogy with linear image processing

## Max-plus convolution

A dilation $\delta$ that is translation invariant and verifies $\delta(\lambda+f)=\lambda+\delta(f)$ for any image $f$ and any constant $\lambda$, 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 $b: \mathbb{Z}^{2} \rightarrow \overline{\mathbb{R}}$ is called structuring function or structuring element.


## Analogy with linear image processing

## Min-plus convolution

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

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



## Analogy with linear image processing

## Vector spaces

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

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

## Complete lattices

A translation invariant dilation is a max-plus convolution with a structurng 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 structurng element $b$

$$
\varepsilon f(x)=\bigwedge f(y)+b(x-y)
$$

## Morphological operators: dilations and erosions

Let $(\mathcal{L}, \bigvee, \Lambda)$ and $\left(\mathcal{L}^{\prime}, \bigvee^{\prime}, \Lambda^{\prime}\right)$ be two complete lattices.

## Dilation

$\delta: \mathcal{L} \rightarrow \mathcal{L}^{\prime}$ is a dilation if and only if for any family $\left(x_{i}\right)_{i \in 1} \subseteq \mathcal{L}$,

$$
\delta\left(\bigvee_{i} x_{i}\right)=\bigvee_{i}^{\prime} \delta\left(x_{i}\right)
$$

## Erosion

$\varepsilon: \mathcal{L} \rightarrow \mathcal{L}^{\prime}$ is an erosion if and only if for any family $\left(x_{i}\right)_{i \in I} \subseteq \mathcal{L}$,

$$
\varepsilon\left(\bigwedge_{i} x_{i}\right)=\bigwedge_{i}^{\prime} \varepsilon\left(x_{i}\right) .
$$

## Morphological operators: openings and closings

From dilations and erosions, many morphorlogical 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: $\gamma \leq i d$
- 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: $\varphi \circ \varphi=\varphi$.


## Morphological operators: adjunction

## Adjunction

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

$$
\delta(x) \leq y \Longleftrightarrow x \leq \varepsilon(y)
$$

## Properties:

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

$$
\forall y \in \mathcal{L}, \varepsilon(y)=\bigvee\left\{x \in \mathcal{L}^{\prime}, \delta(x) \leq y\right\}
$$

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

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

- If $(\varepsilon, \delta)$ 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 fonction $b: \mathbb{Z}^{2} \rightarrow[-255,0] \cap \mathbb{N}$ such that for any $x \in E, \bigvee_{y \in E} b(y-x)=0$, the operators $\delta_{b}$ and $\varepsilon_{b}$ 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:


## Morphological operators: examples on images

Cross flat structuring element:


Morphological operators: examples on images

Quadratic structuring element:

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


Morphological operators: examples on images

Quadratic structuring element:

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


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.


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

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

## Learning problem

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

The dilation $\delta$ with structuring element $W$ is seen as the mapping

$$
\delta_{W}: X \mapsto \tilde{X} \nabla W
$$

defined by

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

## Learning problem:

Given the input image $X$ and an ouput image $Y$, find $W$ such that $\delta_{W}(X)=Y($ or $\approx Y)$.

## Morphological solution by adjunction

Let $(\varepsilon, \delta)$ 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 $\tilde{X} \boxtimes W$ is also a dilation of $W$, which we may note it $\delta_{\tilde{x}}(W)$.
Its adjoint erosion $\varepsilon_{\tilde{\chi}}: \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{(2 p+1)^{2}}$ is defined by

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

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

$$
\min _{W}\left\|Y-\delta_{W}(X)\right\|^{2}
$$

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

## Least square error solution with gradient descent

Convexity:

$$
\left\|Y-\delta_{W}(X)\right\|^{2}=\sum_{i=1}^{m \times n}\left(f_{i}(W)-Y_{i}\right)^{2}
$$

with $f_{i}(W)=(\tilde{X} \nabla W)_{i}=\bigvee_{1 \leq l \leq(2 p+1)^{2}} \tilde{X}_{i, l}+W_{l}$.

## Least square error solution with gradient descent



Left: An example of $f_{i}(W)$ in 2D. Right: $\left(f_{i}(W)-Y_{i}\right)^{2}$ 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

$$
\delta: w \in \mathbb{R}^{n} \mapsto \bigvee_{1 \leq i \leq n}\left\{x_{i}+w_{i}\right\}
$$

is differentiable almost everywhere with respect to w.

## Least square error solution with gradient descent

## Gradient:



## Letting

$E_{j}=\left\{w \in \mathbb{R}^{n}, x_{j}+w_{j}=\bigvee_{1 \leq i \leq n}\left\{x_{i}+w_{i}\right\}\right\}$,
then $\delta$ is differentiable on the interior of $E_{j}$ and

$$
\forall \mathrm{w} \in{\stackrel{\circ}{E_{j}}, \quad \nabla_{\mathrm{w}} \delta(\mathrm{w}, \mathrm{x})=e_{j},{ }^{2} .}
$$

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

## Least square error solution with gradient descent

## Gradient:



Besides, let $I \subseteq\{1, \ldots, n\}$. Then on $\bigcap_{i \in 1} E_{i} \backslash\left(\bigcup_{j \notin \mid} E_{j}\right)$

$$
d_{l}=-\sum_{i \in I} e_{i}
$$

is a descent direction of $\delta$.

## Least square error solution with gradient descent

## Gradient:

For our problem, noting $\ell(W)=\left\|Y-\delta_{W}(X)\right\|^{2}$, we have

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

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

$$
G_{i, k}=\left\{\begin{array}{cc}
1 & \text { if } \tilde{X}_{i k}+W_{k}=\bigvee_{1 \leq 1 \leq(2 p+1)^{2}} \tilde{x}_{i, l}+W_{l} \\
0 & \text { otherwise }
\end{array}\right.
$$

## Learning an opening structuring element

The opening $\gamma$ with structuring element $W$ is seen as the mapping

$$
\gamma_{w}: X \mapsto \delta_{w}\left(\varepsilon_{W}(X)\right)
$$

with

$$
\varepsilon_{W}(X)_{i}=\left(\tilde{X} \boxtimes W^{*}\right)_{i}=\bigwedge_{1 \leq l \leq(2 p+1)^{2}} \tilde{X}_{i, l}-W_{l}^{\top}, \quad 1 \leq i \leq m \times n
$$

and as before

$$
\delta_{W}(Y)_{i}=(\tilde{Y} \nabla W)_{i}=\bigvee_{1 \leq \leq \leq(2 p+1)^{2}} \tilde{Y}_{i, l}+W_{l}, \quad 1 \leq i \leq m \times n
$$

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

## Learning problem:

Given the input image $X$ and an ouput 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 ouput image $Z$, find $W$ such that $\gamma_{w}(X)=Z$ (or $\approx Z$ ).
The problem can also be formulated with an auxiliary variable $Y$ :

$$
\left\{\begin{array}{c}
\tilde{Y} \nabla W=Z \\
\tilde{X} \boxtimes W^{*}=Y
\end{array}\right.
$$

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 paramterized by one structuring element W

$$
\gamma_{W}(X)=\delta_{W}\left(\varepsilon_{W}(X)\right)
$$

- The two layers architecture: an erosion layer followed by a dilation layer with independent parameters $W_{1}$ and $W_{2}$

$$
g_{\underline{w}}(X)=\delta_{W_{2}}\left(\varepsilon_{W_{1}}(X)\right)
$$

where $\underline{W}=\left[W_{1}, W_{2}\right]$.

## Learning an opening structuring element with gradient descent

## Two alternative gradients!

Noting:

- $\partial_{w}$ : derivative with respect to the parameter
- $\partial_{x}$ : derivative with respect to the input
- $\ell$ : loss function, typically $\ell(\hat{Y})=\|\hat{Y}-Y\|_{2}^{2}$ for a fixed $Y$ (label associated to $X$ )
- $\Gamma(W, X)=\ell\left(\gamma_{w}(X)\right) \in \mathbb{R}$
- $G(\underline{W}, X)=\ell\left(g_{\underline{w}}(X)\right) \in \mathbb{R}$
we get

$$
\begin{gathered}
\nabla_{W} \Gamma(W, X)=\nabla \ell\left(\gamma_{W}(X)\right) \cdot\left[\partial_{w} \delta_{W}\left(\varepsilon_{W}(X)\right)+\partial_{x} \delta_{W}\left(\varepsilon_{W}(X)\right) \cdot \partial_{W} \varepsilon_{W}(X)\right]^{\top} \\
\nabla_{W} G(\underline{W}, X)=\nabla \ell(g(\underline{W}, X)) \cdot\left[\begin{array}{c}
\partial_{W} \delta_{W_{2}}\left(\varepsilon_{W_{1}}(X)\right) \\
\partial_{X} \delta_{W_{2}}\left(\varepsilon_{W_{1}}(X)\right) \cdot \partial_{w} \varepsilon_{W_{1}}(X)
\end{array}\right]^{\top}
\end{gathered}
$$

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) K of dimensions $k_{1} \times k_{2}$, the cross-correlation operation is given by:

$$
\begin{equation*}
(\mathrm{I} * \mathrm{~K})_{i j}:=\sum_{m=0}^{k_{1}-1} \sum_{n=0}^{k_{2}-1} \mathrm{I}(i+m, j+n) \mathrm{K}(m, n), \tag{3}
\end{equation*}
$$

$\forall i \in 1, \ldots, M$ and $j \in 1, \ldots, N$ Note that $(1 * K)_{i j}$ 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

## Leaning morphological layers

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 I, a bank of filters 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 $K \in \mathbb{R}^{k_{1} \times k_{2} \times C \times D}$ and biases $\mathrm{b} \in \mathbb{R}^{D}$, one for each filter.

The output from this convolution procedure is as follows:

$$
\begin{equation*}
\left[(\mathbf{I} * \mathrm{~K})_{i j}\right]_{d}:=\sum_{c=1}^{c} \sum_{m=0}^{k_{1}-1} \sum_{n=0}^{k_{2}-1} \mathrm{I}(i+m, j+n, c) \mathrm{K}(m, n, c, d)+\mathrm{b}(d) \tag{4}
\end{equation*}
$$

$\forall i \in 1, \ldots, M$ and $j \in 1, \ldots, N$. Note that $\left[(I * K)_{i j}\right]_{d}$ 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



## 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 $\left(s_{1}, s_{2}\right)$, is defined as $[(I * K) i j]_{d}, \forall i \in s_{1}-1,2 s_{1}-1,3 s_{1}-1, \ldots, M$ and $\forall j \in s_{2}-1,2 s_{2}-1,3 s_{2}-1 \ldots, N$.

Image

| 1 | 2 | 5 | 0 |
| :---: | :---: | :---: | :---: |
| 0 | 3 | 1 | 2 |
| -1 | 3 | 4 | 2 |
| 0 | 6 | 7 | 8 |

Kernel


Convolution

| 0 | 2 | 4 | 10 |
| :---: | :---: | :---: | :---: |
| 1 | 3 | 13 | 7 |
| 0 | 1 | 10 | 11 |
| -1 | 2 | 19 | 20 |

Convolution with strides $(2,2)$

| 0 | 4 |
| :---: | :---: |
| 0 | 10 |

Figure 5: Example of a convolution with strides (2,2)
$\left[(1 * K)_{i j}\right]_{d}$ with strides $\left(s_{1}, s_{2}\right)$ 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)

$$
\begin{array}{r}
\text { MaxPooling }\left[l_{i j}\right]_{c}:=\bigvee_{m=0}^{k_{1}-1} \bigvee_{n=0}^{k_{2}-1} I(i+m, j+n, c) \\
\text { AveragePooling[}\left[l_{i j}\right]_{c}:=\sum_{m=0}^{k_{1}-1} \sum_{n=0}^{k_{2}-1} 1(i+m, j+n, c) /\left(k_{1} * k_{2}\right)  \tag{6}\\
\forall i \in s_{1}-1,2 s_{1}-1,3 s_{1}-1, \ldots, M \text { and } \forall j \in s_{2}-1,2 s_{2}-1,3 s_{2}-1 \ldots, N
\end{array}
$$

## 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)

$$
\begin{array}{r}
\text { MaxPooling }\left[l_{i j}\right]_{c}:=\bigvee_{m=0}^{k_{1}-1} \bigvee_{n=0}^{k_{2}-1} I(i+m, j+n, c) \\
\text { AveragePooling[}\left[l_{i j}\right]_{c}:=\sum_{m=0}^{k_{1}-1} \sum_{n=0}^{k_{2}-1} 1(i+m, j+n, c) /\left(k_{1} * k_{2}\right)  \tag{6}\\
\forall i \in s_{1}-1,2 s_{1}-1,3 s_{1}-1, \ldots, M \text { and } \forall j \in s_{2}-1,2 s_{2}-1,3 s_{2}-1 \ldots, N
\end{array}
$$

## Example Pooling

Image

| 4 | 2 | 6 | 0 |
| :--- | :--- | :--- | :--- |
| 2 | 4 | 4 | 2 |
| 0 | 4 | 6 | 2 |
| 0 | 6 | 8 | 8 |

Max Pooling strides(1,1)

| 4 | 4 | 6 | 6 |
| :--- | :--- | :--- | :--- |
| 4 | 4 | 6 | 6 |
| 2 | 4 | 6 | 6 |
| 0 | 6 | 8 | 8 |

Max Pooling strides(2,2)

| 4 | 6 |
| :--- | :--- |
| 2 | 6 |

Average Pooling strides(1,1) Average Pooling strides( 2,2 )

| 1 | 1.5 | 2 | 1.5 |
| :---: | :---: | :---: | :---: |
| 1.5 | 3 | 4 | 3 |
| 0.5 | 2.5 | 4.5 | 3.5 |
| 0 | 2.5 | 6 | 6 |


| 1 | 2 |
| :---: | :---: |
| 0.5 | 4.5 |

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

- Note that Pooling $\left[(1 * K)_{i j}\right]_{d}$ with strides $\left(s_{1}, s_{2}\right)$ 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


## Max-Pooling layers as dilations

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 <br> Equivariance 

Equivariance and Invariance

## Invariance



- A property that is perserved 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 $\psi$ is invariant to a set of transformations $\left\{L_{g} \mid g \in G\right\}$ 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

$$
\begin{aligned}
& \text { A tuple }(G, \cdot) \text { is a group if } \\
& \text { 1. } \forall g, h, k \in G, \quad(g \cdot h) \cdot k=g \cdot(h \cdot k) \text {; } \\
& \text { 2. } \exists e \in G, \forall g \in G \quad, e \cdot g=g \cdot e=g \text {; } \\
& \text { 3. } \forall g \in G, \exists g^{-1} \quad, g^{-1} \cdot g=g \cdot g^{-1}=e
\end{aligned}
$$

- 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 $L_{g} \circ L_{h}=L_{h \cdot g}$
- Rotation:

- Translation:
- Scaling:

- Scaling and translation:
- Occlusion(semigroup):

- Shear:



## Equivariance



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

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

where $R_{x}(I)(y)=I(x+y)$. We have $R_{z}(I) \star h=R_{z}(I \star h)$.

## Equivariance Formal Definition

Given an operator $\psi: X \rightarrow Y$ and a (semi)group $G$, then we say that $\psi$ is equivariant with respect to $G$ if there exists two actions $L_{g}, L_{g}^{\prime}, g \in G$, on $X$ and $Y$, respectively, such that, for all $I \in X$

$$
\psi\left[L_{g}(I)\right]=L_{g}^{\prime}[\psi(I)]
$$

- 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) 1
(b) $\delta(1)-1$
(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: <br> Introduction to Invariance and <br> <br> Equivariance 

 <br> <br> Equivariance}

Morphological Scale-Spaces

## Re-Scaling

- The action of scaling can be written as $\left(R_{s}^{\prime} f\right)(x)=f\left(s^{-1} x\right)$
- 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]

$$
\left(I \star_{G} h\right)\left(2^{-k}, x\right)=\sum_{l \geq 0} \sum_{y \in \mathbb{Z}^{2}} 1\left(2^{-k-1}, 2^{k} y+x\right) h\left(2^{-l}, y\right)
$$



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


## Scale-Spaces

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

$$
T_{t}\left[R_{s}^{\prime}(I)\right]=R_{s}^{\prime}\left[T_{\frac{t}{s^{p}}}(I)\right]
$$

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

t
- We have $T_{\mathcal{G}_{t}} \circ R_{S}^{\prime}=R_{S}^{\prime} \circ T_{\mathcal{G}_{t / s^{2}}}$


## Scale-Spaces

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

$$
\left\{\begin{array}{l}
u_{t}=\operatorname{div}(\nabla u)=u_{x x}+u_{y y} \\
u(0, x, y)=f(x, y)
\end{array}\right.
$$

- Morphological scale-spaces arise from the Hamilton-Jacobi equation $u_{t}=H(x, \nabla u)$
- Quadratic morphological scale-spaces come from

$$
\left\{\begin{array}{l}
u_{t}= \pm\|\nabla u\|^{2} \\
u(0, x, y)=f(x, y)
\end{array}\right.
$$

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

## Quadratic Morphological Scale-Spaces

Quadratic Dilations and Erosions

$$
\begin{aligned}
& \delta_{t}(I)(x)=\bigvee_{y \in \mathbb{R}^{d}} I(x-y)-c \cdot \frac{\|y\|^{2}}{4 t} \\
& \varepsilon_{t}(I)(x)=\bigwedge_{y \in \mathbb{R}^{d}} I(x+y)+c \cdot \frac{\|y\|^{2}}{4 t}
\end{aligned}
$$

The parameter c is optimized along with the weights of the network.
Quadratic Openings and Closings

$$
\begin{aligned}
\gamma_{\mathrm{t}} & =\delta_{\mathrm{t}} \circ \varepsilon_{\mathrm{t}}, \\
\varphi_{\mathrm{t}} & =\varepsilon_{\mathrm{t}} \circ \delta_{\mathrm{t}} .
\end{aligned}
$$

## Properties

- Commutation with scaling:

$$
\begin{aligned}
& \delta_{t} \circ R_{s}^{\prime}=R_{s}^{\prime} \circ \delta_{t / s^{2}} \\
& \varepsilon_{t} \circ R_{s}^{\prime}=R_{s}^{\prime} \circ \varepsilon_{t / s^{2}} \\
& \gamma_{t} \circ R_{s}^{\prime}=R_{s}^{\prime} \circ \gamma_{t / s^{2}} \\
& \varphi_{t} \circ R_{s}^{\prime}=R_{\mathrm{s}}^{\prime} \circ \varphi_{t / s^{2}}
\end{aligned}
$$

- Composition:

$$
\begin{aligned}
\delta_{t} \circ \delta_{s} & =\delta_{t+s} \\
\varepsilon_{t} \circ \varepsilon_{s} & =\varepsilon_{t+s} \\
\gamma_{t} \circ \gamma_{s} & =\gamma_{t \vee s} \\
\varphi_{t} \circ \varphi_{s} & =\varphi_{t \vee s}
\end{aligned}
$$

## Quadratic Dilations and Erosions


t

## Quadratic Closings and Openings


t

## 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

## References i

(-i 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.
國 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.

## References ii

國 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.
圊 Karaev, S. and Miettinen, P. (2019).
Algorithms for approximate subtropical matrix factorization.
Data Mining and Knowledge Discovery, 33(2):526-576.

## References iii

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.
囯 Maragos, P., Charisopoulos, V., and Theodosis, E. (2021).
Tropical geometry and machine learning.
Proceedings of the IEEE, 109(5):728-755.
囯 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.

## References iv

(围 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.
Rangalli, 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.

## References v

Tilivis, 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.
囯 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.

