

Math and Images

Having fun with your own image filter

Stefano Gualandi

Università di Pavia, Dipartimento di Matematica

email: stefano.gualandi@unipv.it

twitter: [@famo2spaghi](https://twitter.com/famo2spaghi)

blog: <http://stegua.github.com>

web: <http://matematica.unipv.it/gualandi/>

Math and Images

Disclaimer: "Images are just numbers!"

Math and Images

Disclaimer: "Images are just numbers!"



64x64 pixels

MedNIST Challenge

Disclaimer: "Images are just numbers!"



64x64 pixels
MedNIST Challenge



16x16 pixels
Downsampling

Math and Images

Disclaimer: "Images are just numbers!"



64x64 pixels
MedNIST Challenge



16x16 pixels
Downsampling

17	14	13	12	15	14	14	87	82	16	14	14	13	14	16	21
15	12	11	10	104	77	10	123	116	23	101	13	10	12	14	19
13	10	10	11	115	108	13	146	146	63	117	29	8	10	11	17
12	9	90	64	106	140	11	159	145	114	148	21	6	7	10	16
11	7	99	123	85	152	98	184	184	165	161	17	8	8	13	16
11	6	44	135	110	166	152	174	177	168	171	5	8	93	114	15
11	6	5	139	162	153	185	195	189	196	163	5	102	171	139	16
10	5	6	46	161	168	208	218	214	227	177	6	165	173	54	15
9	4	25	22	170	188	212	210	210	230	170	158	207	156	6	11
8	3	241	147	119	217	224	225	225	244	182	215	214	86	6	11
5	1	91	249	228	211	247	253	253	254	234	216	158	5	5	11
5	1	16	186	81	195	255	255	255	255	252	152	5	4	4	11
6	2	1	1	3	234	255	255	255	253	138	4	0	1	3	10
6	2	2	2	5	255	255	255	255	218	27	5	1	2	3	10
6	3	2	2	95	255	251	255	255	115	2	2	1	1	3	10
8	4	4	3	171	255	255	255	255	5	3	3	3	3	4	10

Image as matrix of
numbers

Math, Color Images, and Tensors



RGB Image

Math, Color Images, and Tensors



RGB Image



Red channel



Green channel



Blu channel

Images and Point Clouds

RGB Image

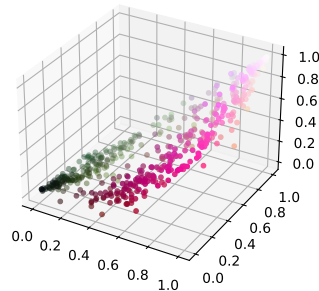


Images and Point Clouds

RGB Image



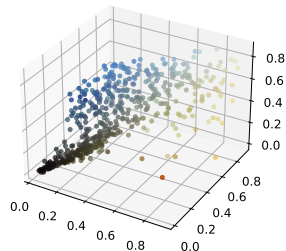
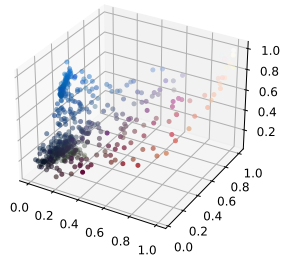
Sampling of pixels by colors in 3D



How to generate a fake van Gogh for Borgo Ticino

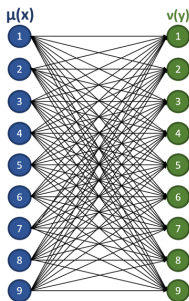


How to generate a fake van Gogh for Borgo Ticino



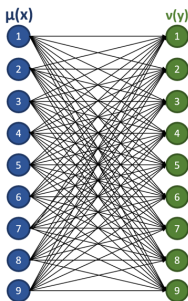
Computing the KWD between two spatial distributions

Given two empirical distributions μ and ν defined over a space X , we can define a complete bipartite graph:



Computing the KWD between two spatial distributions

Given two empirical distributions μ and ν defined over a space X , we can define a complete bipartite graph:



Using the bipartite graph $B = (V \cup W, E)$, we can write the LP *bipartite* model:

$$\min \sum_{ij \in E} c_{ij} \pi_{ij} \quad (1)$$

$$\text{s.t.} \quad \sum_{ij \in E} \pi_{ij} = \mu_i, \quad \forall i \in V, \quad (2)$$

$$\sum_{ij \in E} \pi_{ij} = \nu_j, \quad \forall j \in W, \quad (3)$$

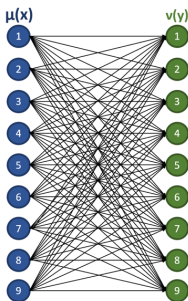
$$\pi_{ij} \geq 0, \quad \forall \{i, j\} \in E. \quad (4)$$

where the cost are defined as:

$$c(i, j) := d(x_i, x_j) = \|x_i - x_j\|_2. \quad (5)$$

Computing the KWD between two spatial distributions

Given two empirical distributions μ and ν defined over a space X , we can define a complete bipartite graph:



Using the bipartite graph $B = (V \cup W, E)$, we can write the LP *bipartite* model:

$$\min \sum_{ij \in E} c_{ij} \pi_{ij} \quad (1)$$

$$\text{s.t.} \quad \sum_{ij \in E} \pi_{ij} = \mu_i, \quad \forall i \in V, \quad (2)$$

$$\sum_{ij \in E} \pi_{ij} = \nu_j, \quad \forall j \in W, \quad (3)$$

$$\pi_{ij} \geq 0, \quad \forall \{i, j\} \in E. \quad (4)$$

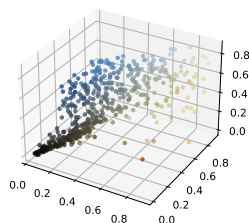
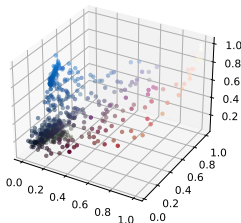
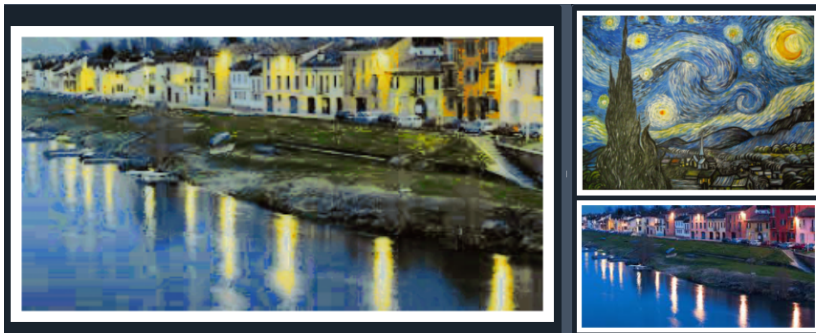
where the cost are defined as:

$$c(i, j) := d(x_i, x_j) = \|x_i - x_j\|_2. \quad (5)$$

REMARK

We need to solve an optimization problem to compute the Kantorovich Wasserstein Distance. We use an optimal transportatino plan to mapping the colors of van Gogh.

How to generate a fake van Gogh for Borgo Ticino



Old Math, New (Snapchat?) Applications



Old Math, New (Snapchat?) Applications



M É M O I R E SUR LA THÉORIE DES DÉBLAIS ET DES REMBLAIS.

Par M. M O N G E.

Lorsqu'on doit transporter des terres d'un lieu dans un autre, on a coutume de donner le nom de *Déblai* au volume des terres que l'on doit transporter, & le nom de *Remblai* à l'espace qu'elles doivent occuper après le transport.

Le prix du transport d'une molécule étant, toutes choses d'ailleurs égales, proportionnel à son poids & à l'espace qu'on lui fait parcourir, & par conséquent le prix du transport total devant être proportionnel à la somme des produits des molécules multipliées chacune par l'espace parcouru, il s'ensuit que le déblai & le remblai étant donnés de figure & de position, il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, d'après laquelle la somme de ces produits sera la moindre possible, & le prix du transport total

Old Math, New (Snapchat?) Applications

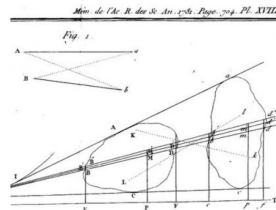


M É M O I R E S U R L A T H É O R I E D E S D É B L A I S E T D E S R E M B L A I S.

Par M. M O N G E.

Lorsqu'on doit transporter des terres d'un lieu dans un autre, on a coutume de donner le nom de *Déblai* au volume des terres que l'on doit transporter, & le nom de *Remblai* à l'espace qu'elles doivent occuper après le transport.

Le prix du transport d'une molécule étant, toutes choses d'ailleurs égales, proportionnel à son poids & à l'espace qu'on lui fait parcourir, & par conséquent le prix du transport total devant être proportionnel à la somme des produits des molécules multipliées chacune par l'espace parcouru, il s'ensuit que le déblai & le remblai étant donnés de figure & de position, il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, d'après laquelle la somme de ces produits sera la moindre possible, & le prix du transport total



Old Math, New (Snapchat?) Applications



M É M O I R E S U R L A T H É O R I E D E S D É B L A I S E T D E S R E M B L A I S.

Par M. M O N G E.

Lorsqu'on doit transporter des terres d'un lieu dans un autre, on a coutume de donner le nom de *Déblai* au volume des terres que l'on doit transporter, & le nom de *Remblai* à l'espace qu'elles doivent occuper après le transport.

Le prix du transport d'une molécule étant, toutes choses d'ailleurs égales, proportionnel à son poids & à l'espace qu'on lui fait parcourir, & par conséquent le prix du transport total devant être proportionnel à la somme des produits des molécules multipliées chacune par l'espace parcouru, il s'ensuit que le déblai & le remblai étant donnés de figure & de position, il n'est pas indifférent que telle molécule du déblai soit transportée dans tel ou tel autre endroit du remblai, mais qu'il y a une certaine distribution à faire des molécules du premier dans le second, d'après laquelle la somme de ces produits sera la moindre possible, & le prix du transport total

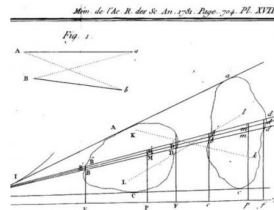


Figure 1. L. V. Kantorovich in his youth.

Honorary Lecture by 2018 Fields medallist Alessio Figalli



<https://www.youtube.com/watch?v=m9b1CEqXAH8>

Outline

1 Introduction

2 Frattali

3 Convolutions

4 Geometric trasformations

Frattali di Julia

1^a Competizione sui Frattali

Studenti del corso di Programmazione 2, A.A. 2016/2017

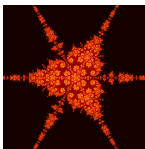
Università degli Studi di Pavia, Dipartimento di Matematica "F. Casorati"



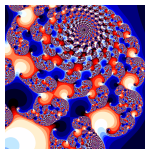
REGOLAMENTO

Scopo della competizione era **programmare** in Python il frattale "più bello", votato dagli stessi studenti, che in aula hanno dato due preferenze: la prima valeva 2 punti, la seconda 1 punto. Il primo classificato ha ricevuto 19 punti e 10 voti; il secondo ha ricevuto 13 punti e 12 voti. Il docente ha dato il "premio alla critica".

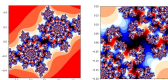
Primo Classificato



Secondo Classificato

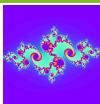


Premio della Critica

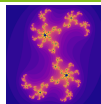


FRATTALI DI JULIA
 $z^2 + 1.5 + 0.8i$

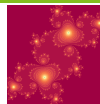
ID 1



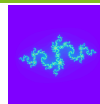
ID 2



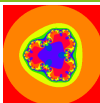
ID 4



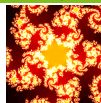
ID 5



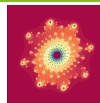
ID 8



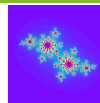
ID 9



ID 11



ID 13



ID 18



ID 20



ID 21



ID 23



Frattali di Julia: Codice Python

```

28 @jit
29 def JuliaSetForIter(z, c, it, maxit=64):
30     """ Semplice processo iterativo """
31     for it in range(maxit):
32         z = z**2 + c
33         if abs(z) > 2:
34             return it
35     return 0
36
37 @jit
38 def JuliaSet(z, c=-0.413):
39     return JuliaSetForIter(z, c, 0)
40
41
42 def PlotFractal(A):
43     """ Stampa il frattale a video e lo salva su un file .pdf """
44     plt.figure(figsize=(5,5))
45     plt.tick_params(
46         axis='both',          # changes apply to the x-axis
47         which='both',        # both major and minor ticks are affected
48         bottom='off',         # ticks along the bottom edge are off
49         top='off',            # ticks along the top edge are off
50         left='off', LabelLeft='off',
51         labelbottom='off') # labels along the bottom edge are off
52
53     # SPECIFICARE IL NOME DEL VOSTRO FRATTALE
54     plt.title("Frattale Flop, by Hans")
55     # SPECIFICARE LA SUCCESSIONE USATA, E IL NUMERO MASSIMO DI ITERAZIONI
56     plt.xlabel('$z_{k+1}=z_k^2 - 0.423$ \naltre info...')
57
58     plt.imshow(A, extent=(-scale*n, scale*n, -scale*n, scale*n), animated=False, cmap='jet')
59     # SALVARE L'IMMAGINE SU UN PDF CON UNA PRECISIONE DPI=300
60     plt.savefig("prova.pdf", dpi=300)
61
62     plt.show()

```

Frattali di Julia: Animazione

DEMO animations

Outline

1 Introduction

2 Frattali

3 Convolutions

4 Geometric trasformations

Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

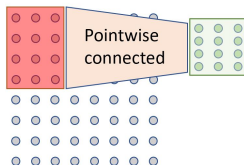
input image



0	0	1
0	0	-1
0	0	-1
0	0	1

*kernel image
(filter)*

*new image
(feature map)*



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

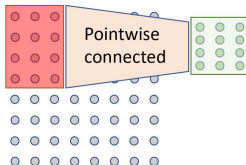
0	0	1
0	0	-1
0	0	-1
0	0	1

=

⊕

=





Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

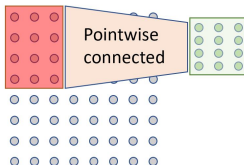
0	0	0
0	0	-1
0	0	0
0	0	0

⊕

=

-1

-1					



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

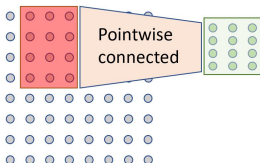
0	0	0
0	0	-1
0	0	0
0	0	0

⊕

=

-1

-1	-1				



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

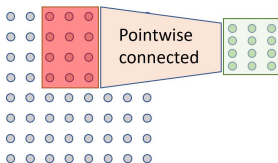
0	0	0
0	0	0
0	0	0
0	0	0

⊕

=

0

-1	-1	0			



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

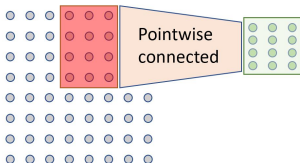
0	0	0
0	0	0
0	0	0
0	0	0

⊕

=

0

-1	-1	0	0		



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

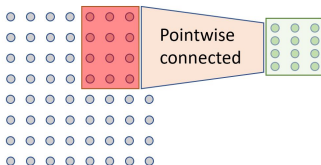
0	0	0
0	0	0
0	0	0
0	0	0

⊕

=

0

-1	-1	0	0	0	



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

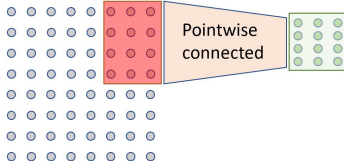
0	0	0
0	0	0
0	0	0
0	0	0

⊕

=

0

-1	-1	0	0	0	0



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

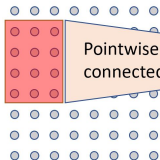
0	0	1
0	0	0
0	0	0
0	0	1

⊕

=

2

-1	-1	0	0	0	0
2					



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

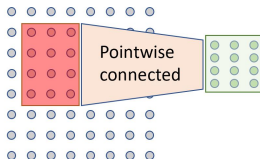
0	0	1
0	0	0
0	0	0
0	0	1

⊕

=

2

-1	-1	0	0	0	0
2	2				



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

⊗

0	0	1
0	0	-1
0	0	-1
0	0	1

=

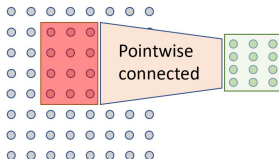
0	0	0
0	0	0
0	0	0
0	0	0

⊕

=

0

-1	-1	0	0	0	0
2	2	0			



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

input image

$\otimes$

0	0	1
0	0	-1
0	0	-1
0	0	1

*kernel image
(filter)*

$=$

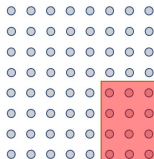
0	0	0
0	0	0
0	0	0
0	0	0

$\oplus$

$=$ 0

-1	-1	0	0	0	0
2	2	0	0	0	0
-1	-1	0	0	0	0
-1	-1	0	0	0	0
1	1	0	0	0	0

*new image
(feature map)*



Pointwise
connected



Convolution

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



0	0	1
0	0	-1
0	0	-1
0	0	1



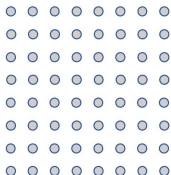


-1	-1	0	0	0	0
2	2	0	0	0	0
-1	-1	0	0	0	0
-1	-1	0	0	0	0
1	1	0	0	0	0

We have performed a **CONVOLUTION** of the *input image* with a *kernel image (filter)*, producing a *new image (feature map)*.

Many important image transformations are convolutions where you *convolve* the image function with a very small "*kernel function*"

input image



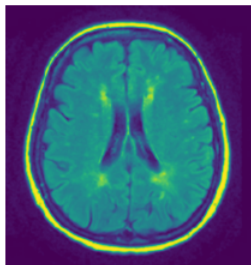
kernel image (filter)



new image (feature map)



Kernels for Edge Detection



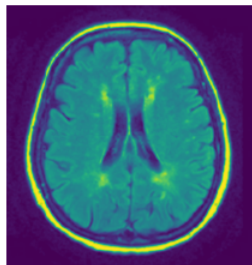
Two kernels/filters

Input image:

Brain MRI

1190x1123 pixels

Kernels for Edge Detection



Input image:

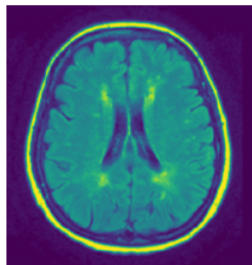
Brain MRI

1190x1123 pixels

Two kernels/filters

$$K_1 = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline +1 & +2 & +1 \\ \hline \end{array}$$

Kernels for Edge Detection



Input image:

Brain MRI

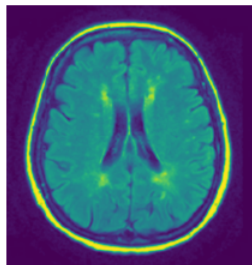
1190x1123 pixels

Two kernels/filters

$$K_1 = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline +1 & +2 & +1 \\ \hline \end{array}$$

$$K_2 = \begin{array}{|c|c|c|} \hline -1 & 0 & +1 \\ \hline -2 & 0 & +2 \\ \hline -1 & 0 & +1 \\ \hline \end{array}$$

Kernels for Edge Detection

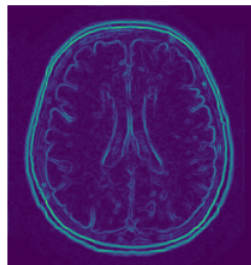


Input image:
Brain MRI
1190x1123 pixels

Two kernels/filters

$$K_1 = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline +1 & +2 & +1 \\ \hline \end{array}$$

$$K_2 = \begin{array}{|c|c|c|} \hline -1 & 0 & +1 \\ \hline -2 & 0 & +2 \\ \hline -1 & 0 & +1 \\ \hline \end{array}$$



Feature Map:
Edge detection

Outline

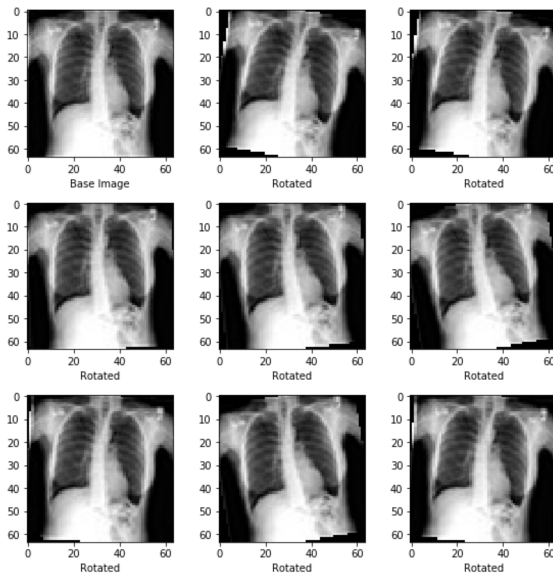
1 Introduction

2 Frattali

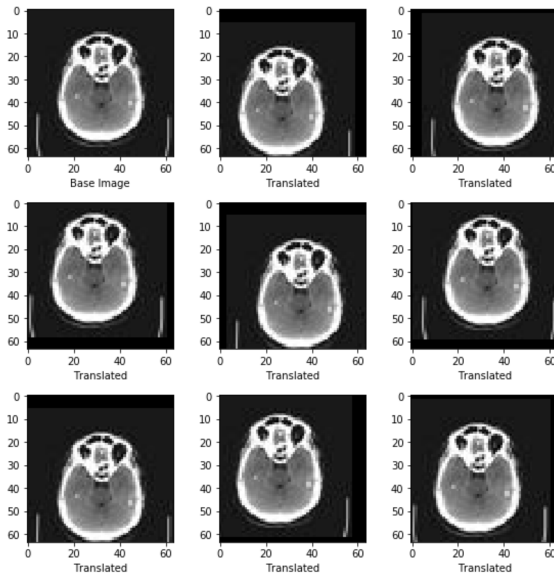
3 Convolutions

4 Geometric trasformations

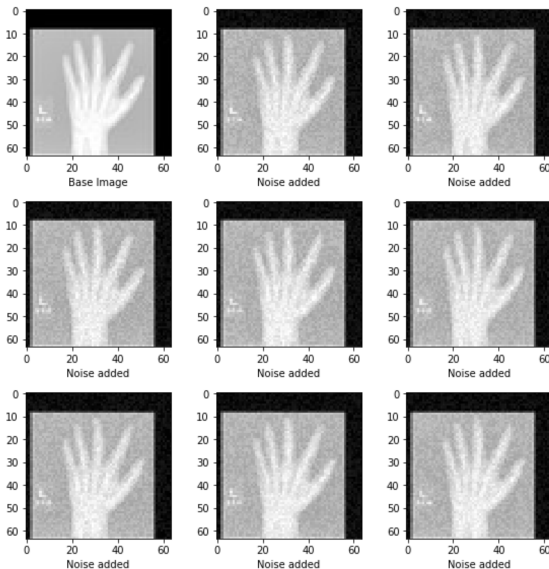
Data Augmentation: Rotations



Data Augmentation: Translation



Data Augmentation: White Noise



Take away message: Math is Fun!



Questions?

