Biomedical Computer Vision

Convolution

October 30th, 2025

Virginia Tasso
virginia.tasso@polimi.it

POLITECNICO MILANO 1863

POLITECNICO MILANO 1863
NECST laboratory

Syllabus:

28.10
- Dimensionality (1D, 2D, 3D, 4D);
- Format (natural jpeg and png, medical MR CT and PET DICOM, Nifti, Tiff).
- Resampling, rescaling, interpolation;
- Histogram, gamma contrast, equalization, clipping, normalization.

30.10
- Convolution, up-convolution, transpose convolution;
- Padding, stride, sliding.

11.11
- Filtering: denoising, sharpening, edge (...)

13.11
- Segmentation: thresholding, masking, k-means, watershed, etc;
- Post-processing: connected components, erosion, dilation.

18.11
- Introduction to DL for Computer Vision
- Supervision: supervised, unsupervised, semi-supervised, self-supervised;
- NN layers;

20.11
- Neural networks training, regularization techniques
- Losses and backpropagation (gradient, ADAM…);

25.11
- Data loading (loading, data augmentation);
- Training (step, epoch, batch size, learning rate, overfitting.
- Image segmentation + U-Net

27.11
- CNN and others for other tasks: detection, restoration, generation (…), review of the state-of-the-art.

Useful Information and Links

Course material:

- notebooks and images are available at the following link: BMCV-repo
- slides and recordings are available here: BMCVCourse

Useful links:

Python: Codecademy (https://www.codecademy.com/), Coursera (Python for everybody Course 1
and 2 https://www.coursera.org/specializations/python#courses)

Convolutions: GitHub (https://github.com/vdumoulin/conv_arithmetic), pdf (https://arxiv.org/pdf/1603.07285). Very useful explanation on youtube (https://www.youtube.com/watch?v=KuXjwB4LzSA&t=281s)

Transposed Convolutions Problems: https://distill.pub/2016/deconv-checkerboard/

Syllabus:

28.10
- Dimensionality (1D, 2D, 3D, 4D);
- Format (natural jpeg and png, medical MR CT and PET DICOM, Nifti, Tiff).
- Resampling, rescaling, interpolation;
- Histogram, gamma contrast, equalization, clipping, normalization.

30.10
- Convolution, up-convolution, transpose convolution;
- Padding, stride, sliding.

11.11
- Filtering: denoising, sharpening, edge (...)

13.11
- Segmentation: thresholding, masking, k-means, watershed, etc;
- Post-processing: connected components, erosion, dilation.

18.11
- Introduction to DL for Computer Vision
- Supervision: supervised, unsupervised, semi-supervised, self-supervised;
- NN layers;

20.11
- Neural networks training, regularization techniques
- Losses and backpropagation (gradient, ADAM…);

25.11
- Data loading (loading, data augmentation);
- Training (step, epoch, batch size, learning rate, overfitting (esempio con stesso batch).
- Image segmentation + U-Net

27.11
- CNN and others for other tasks: detection, restoration, generation (…), review of the state-of-the-art.

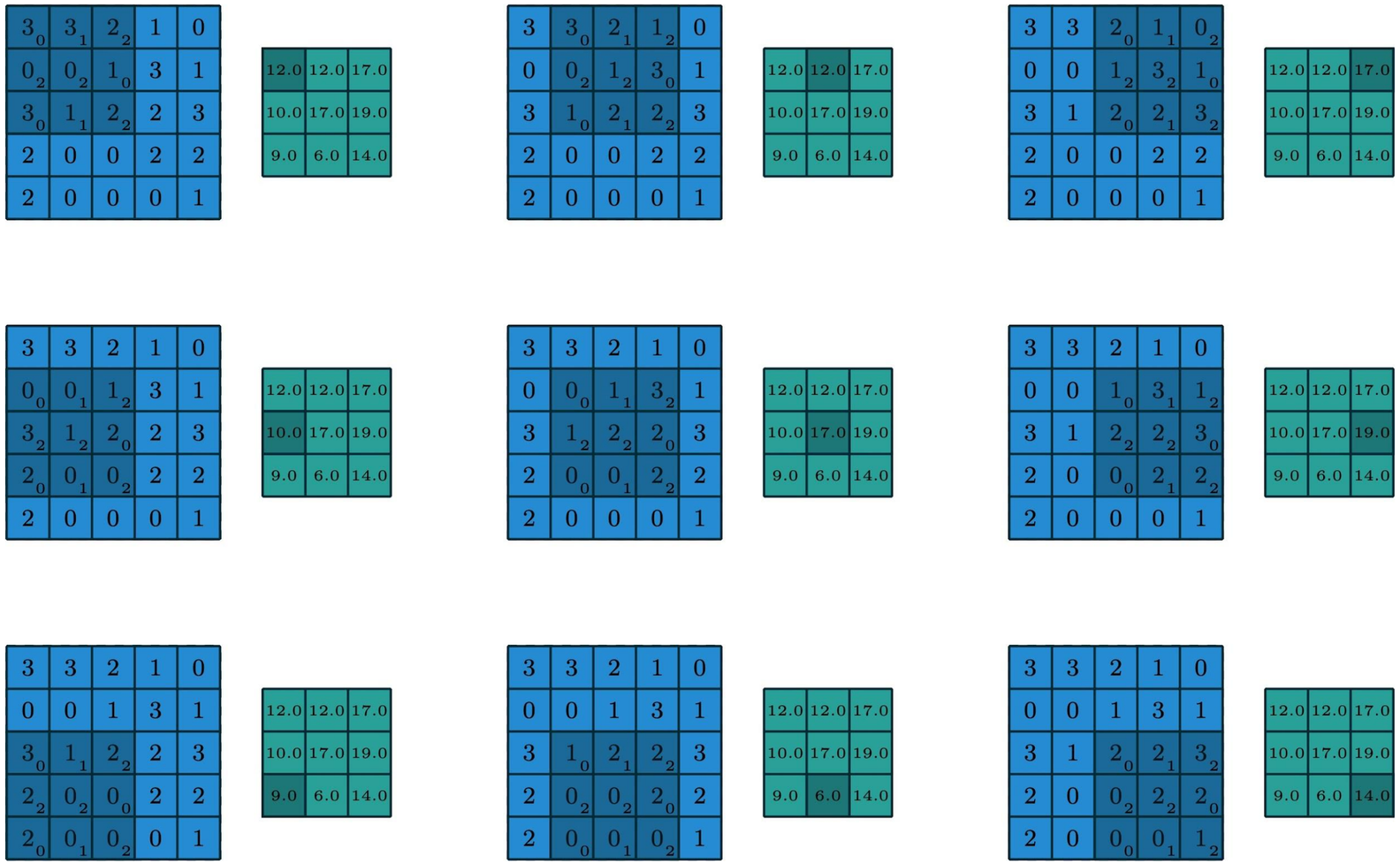
Discrete Convolution

It is a linear transformation that:

- Preserves the notion of ordering
- It is sparse, only few input unites contribute to a given output unit
- Reuses parameters, the same weights are applied to multiple locations in the input

$$(f*g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Discrete Convolution



At each location:

1. Product between each element of the kernel and the input element it overlaps is computed.
2. All the results are summed up.
3. This value is the one of the output feature map

Discrete Convolution

Considering 2D images where the width is direction 1 and the height is dimension 2, important parameters for convolution are:

i_1 *and* i_2 input sizes

k_1 *and* k_2 kernel sizes

$s_1 \equiv s_2$ distance between two consecutive positions of the kernel along the width and height (**stride**)

$p_1 \equiv p_2$ number of zeros concatenated ad the beginning and at the end of each axis (zero padding)

All these parameters affect the output feature map dimensions

Discrete Convolution

Considering 2D images where the width is direction 1 and the height is dimension 2

$i_1 = i_2 = i$ input sizes

$k_1 = k_2 = k$ kernel sizes

$s_1 = s_2 = s$ distance between two consecutive positions of the kernel along the width and height (stride)

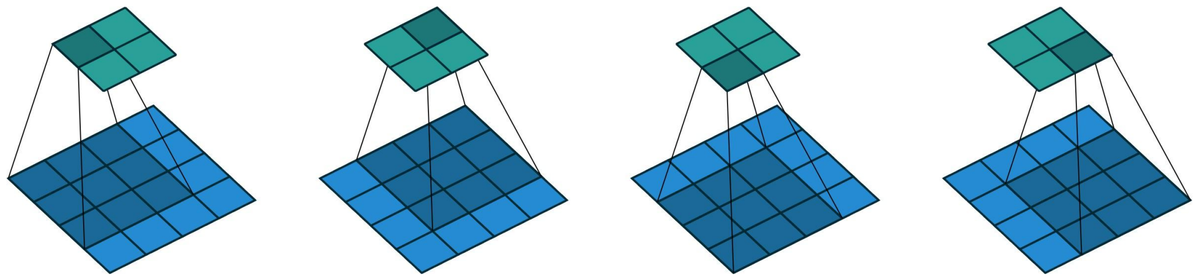
$p_1 = p_2 = p$ number of zeros concatenated ad the beginning and at the end of each axis (zero padding)

Disclaimer: these considerations can be extended to N-D and non-squared cases

Unit Stride

- No Zero Padding s = 1 and p = 0

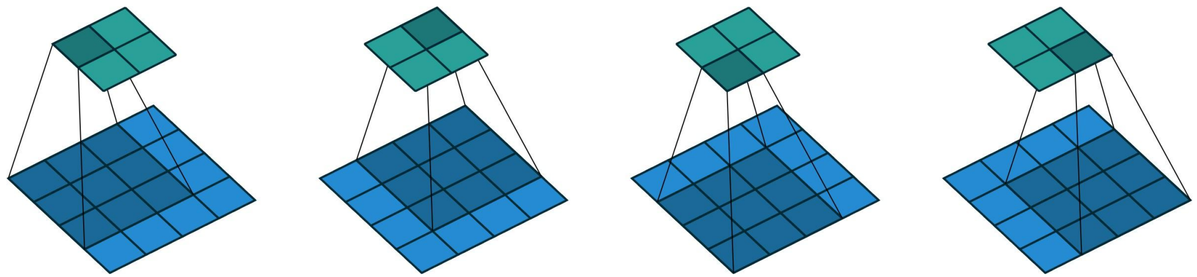
$$o = (i - k) + 1$$



Unit Stride

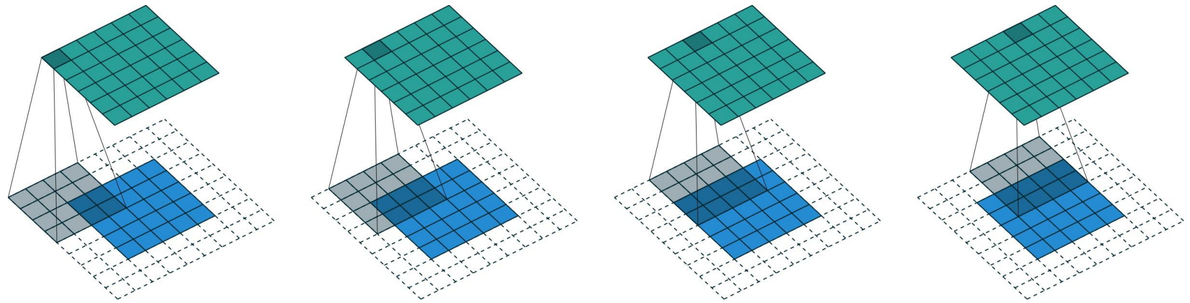
- No Zero Padding s = 1 and p = 0

$$o = (i - k) + 1$$



- Zero Padding

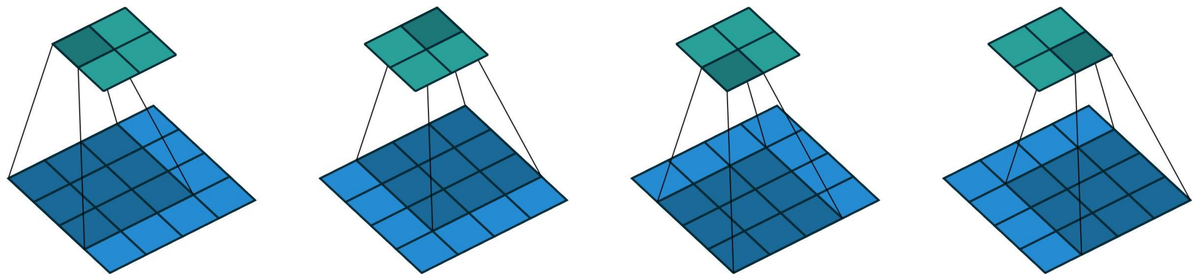
$$o = (i - k) + 2p + 1$$



Unit Stride

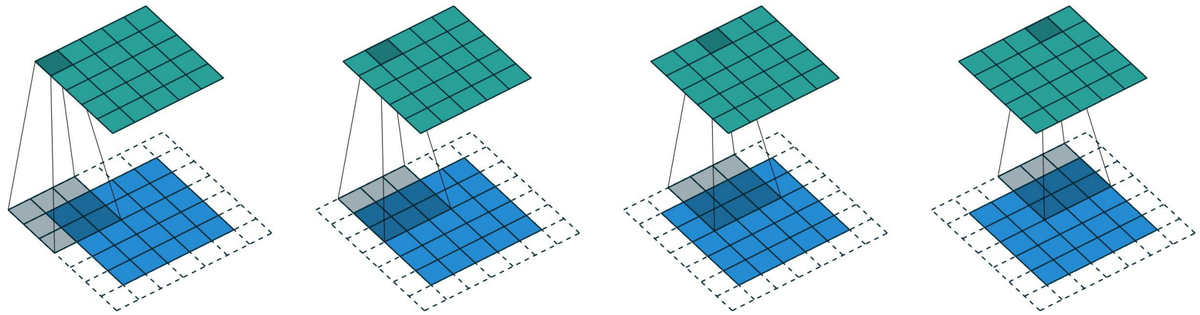
- No Zero Padding s = 1 and p = 0

$$o = (i - k) + 1$$



- Zero Half Padding (same)

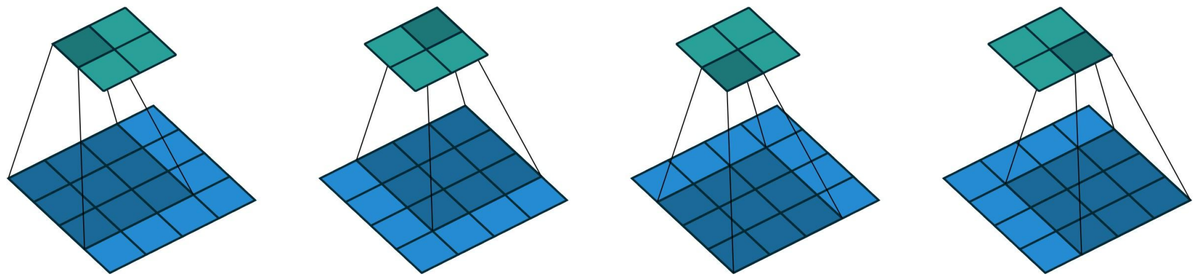
$$o = i + 2\lfloor k/2 \rfloor - (k - 1)$$



Unit Stride

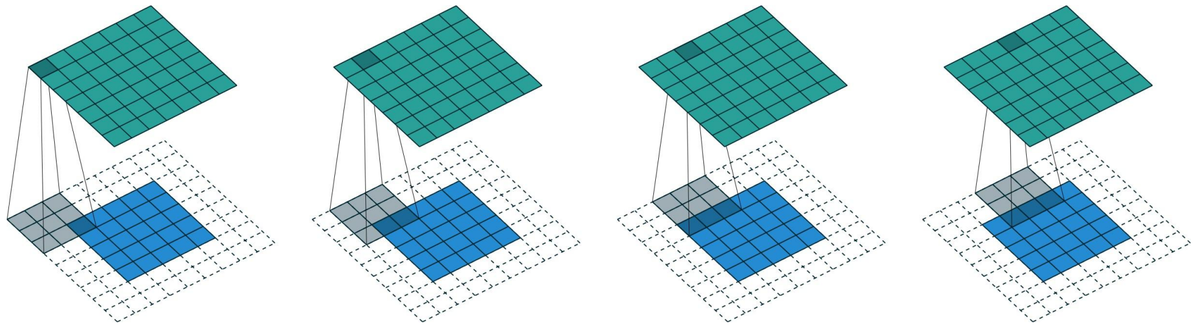
- No Zero Padding s = 1 and p = 0

$$o = (i - k) + 1$$



- Zero Full Padding

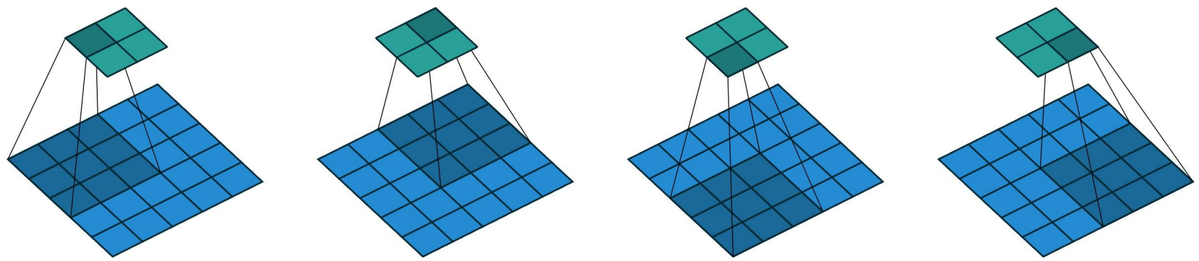
$$o = i + 2(k - 1) - (k - 1)$$



Non-Unit Stride

- No Zero Padding s > 1 and p = 0

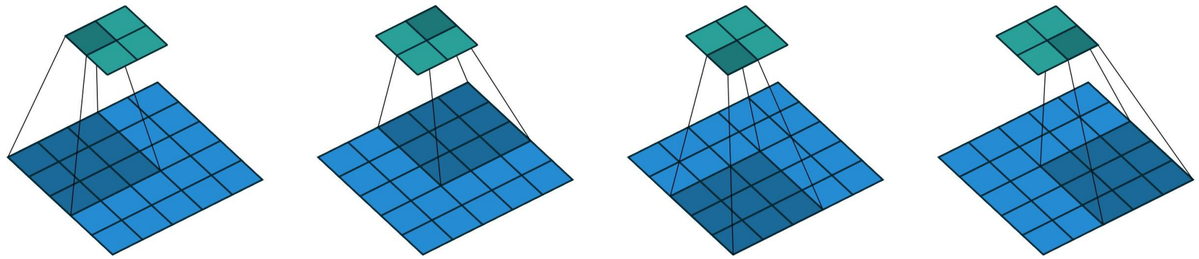
$$o = \left\lfloor \frac{i - k}{s} \right\rfloor + 1$$



Non-Unit Stride

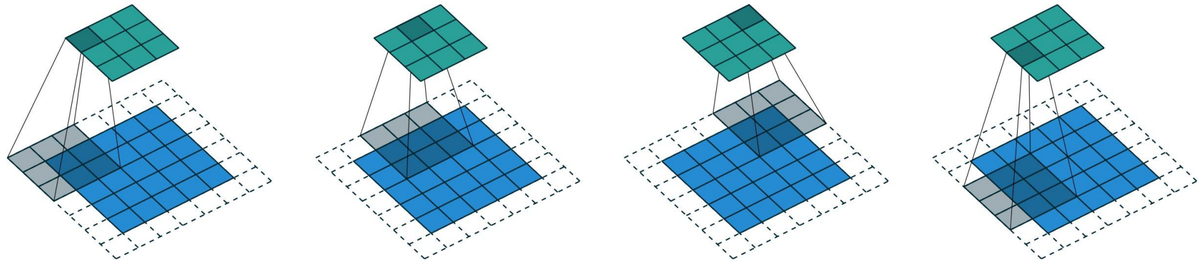
- No Zero Padding s > 1 and p = 0

$$o = \left\lfloor \frac{i - k}{s} \right\rfloor + 1$$



- Zero Padding

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$$



Pooling

Its main characteristics:

- It provides invariance to small translation of the input
- There are different rules (max, min, avg)
- Aggregate input locally by applying non-linearity to the content of some patches

$$o = \left\lfloor \frac{(i-k)}{s} \right\rfloor + 1$$

Transposed Convolution

It is a linear transformation that:

- It goes in the opposite direction of a normal convolution
- Is also known as deconvolution
- It can be always represent as a normal convolution at the cost of a less efficient implementation
- What it does is to recover the initial shape of the feature map

Transposed Convolution

Considering 2D images where the width is direction 1 and the height is dimension 2

i_1 *and* i_2 input sizes

k_1 *and* k_2 kernel sizes

$s_1 \equiv s_2$ distance between two consecutive positions of the kernel along the width and height (stride)

$p_1 \equiv p_2$ number of zeros concatenated ad the beginning and at the end of each axis (zero padding)

All these parameters affect the output feature map dimensions

Transposed Convolution

Considering 2D images where the width is direction 1 and the height is dimension 2

i_1 *and* i_2 input sizes

k_1 *and* k_2 kernel sizes

$s_1 \equiv s_2$ distance between two consecutive positions of the kernel along the width and height (stride)

$p_1 \equiv p_2$ number of zeros concatenated ad the beginning and at the end of each axis (zero padding)

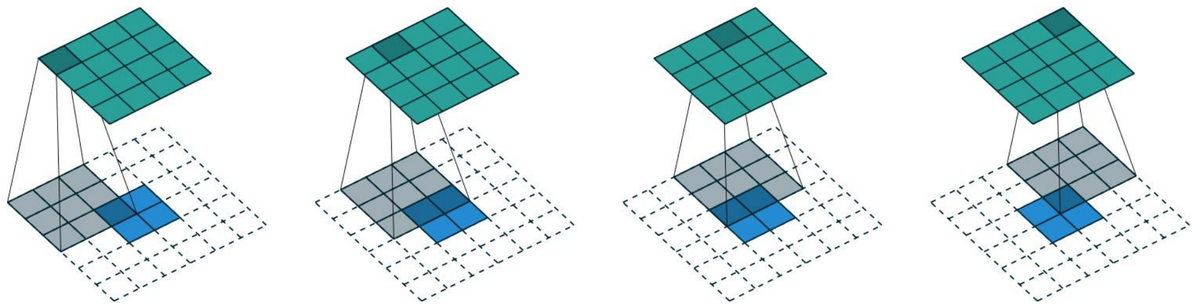
All these parameters affect the output feature map dimensions

Disclaimer: these considerations can be extended to N-D and non-squared cases

Unit Stride

- No Zero Padding s = 1 and p = 0

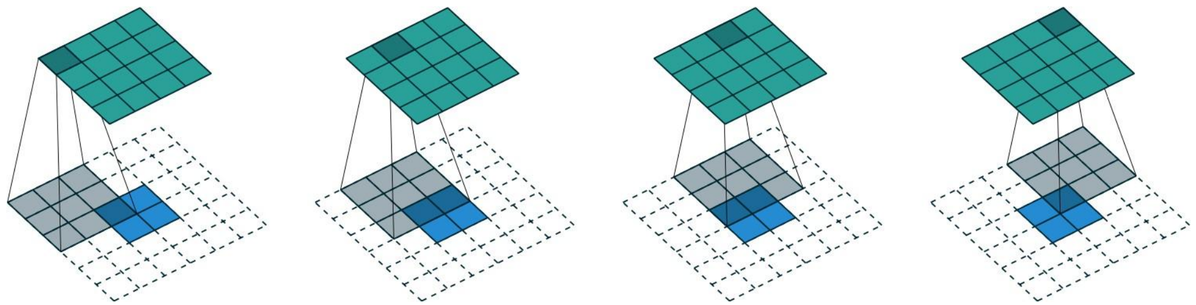
$$o' = (i' + k) - 1$$



Unit Stride

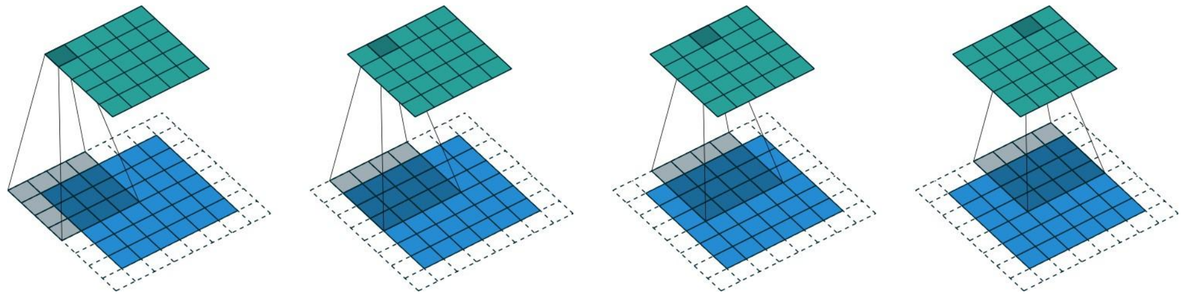
- No Zero Padding s = 1 and p = 0

$$o' = (i' + k) - 1$$



- Zero Padding

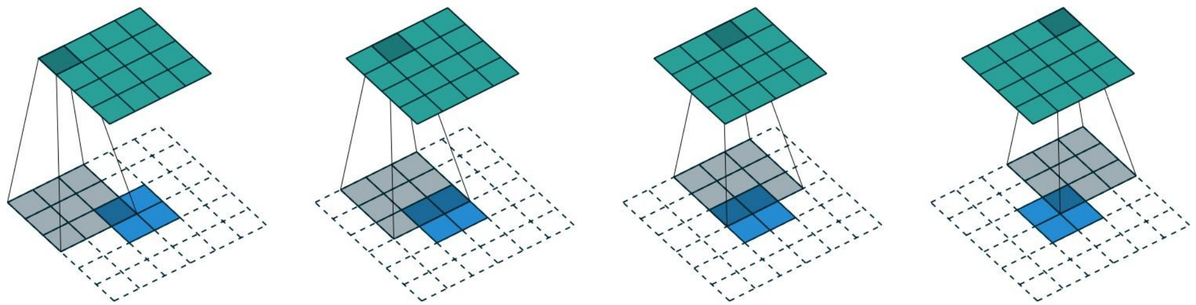
$$o' = i' + (k - 1) - 2p$$



Unit Stride

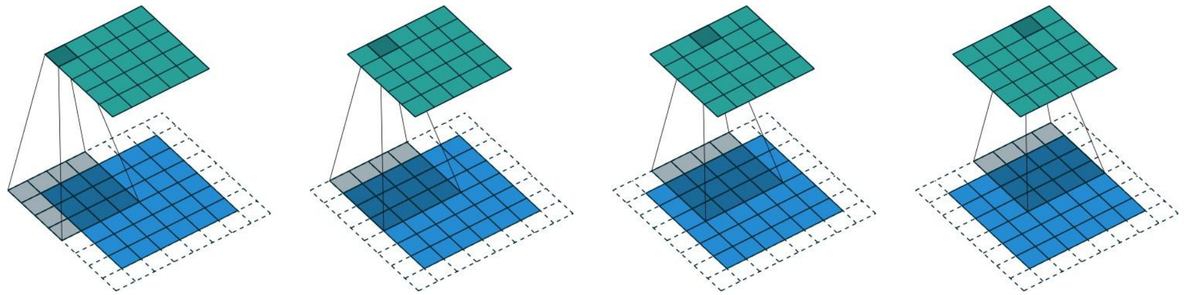
- No Zero Padding s = 1 and p = 0

$$o' = (i' + k) - 1$$



- Zero Half Padding (same)

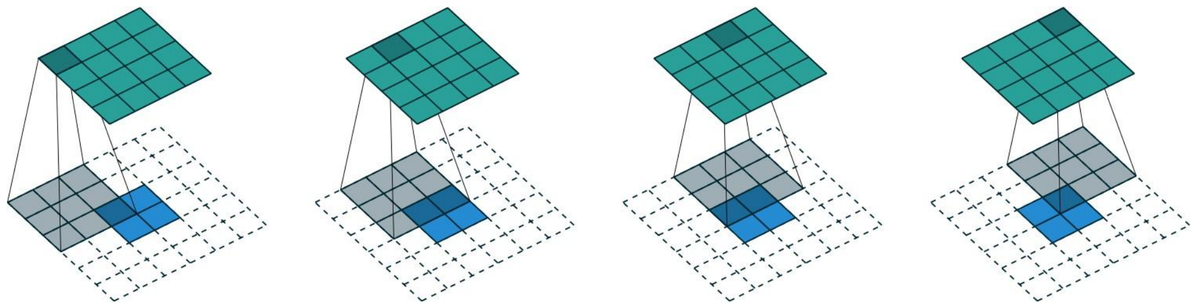
$$o' = i' + (k - 1) - 2\lfloor k / 2 \rfloor$$



Unit Stride

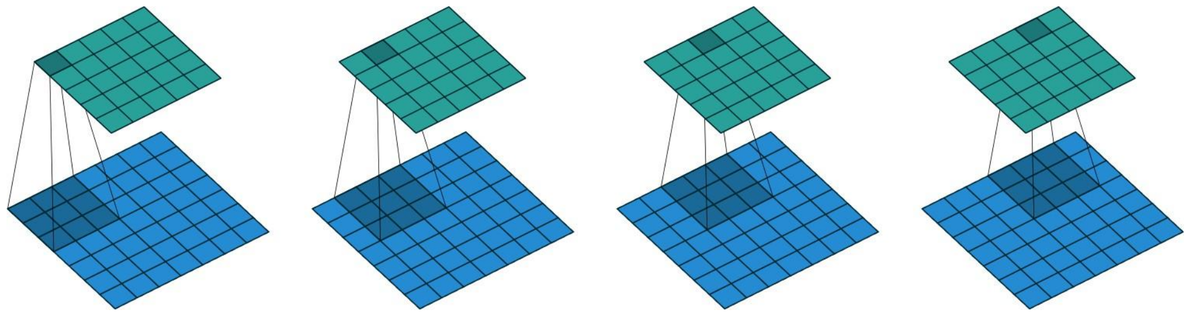
- No Zero Padding s = 1 and p = 0

$$o' = (i' + k) - 1$$



- Zero Full Padding

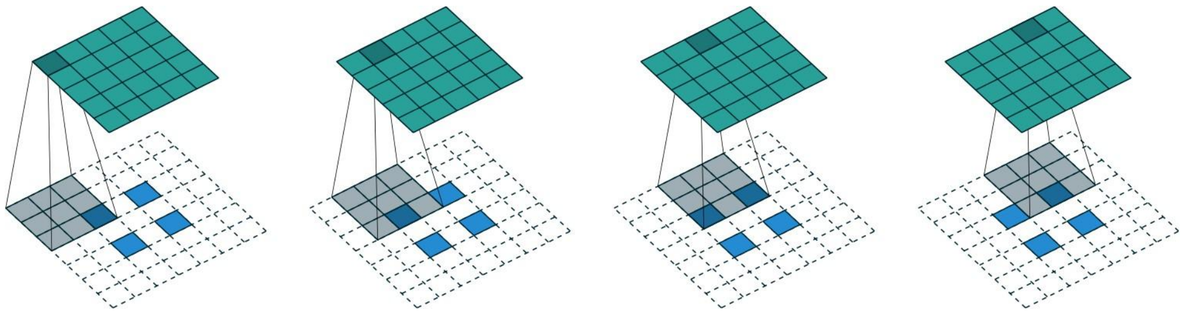
$$o' = i' + (k - 1) - 2p$$



Non-Unit Stride

- No Zero Padding s > 1 and p = 0

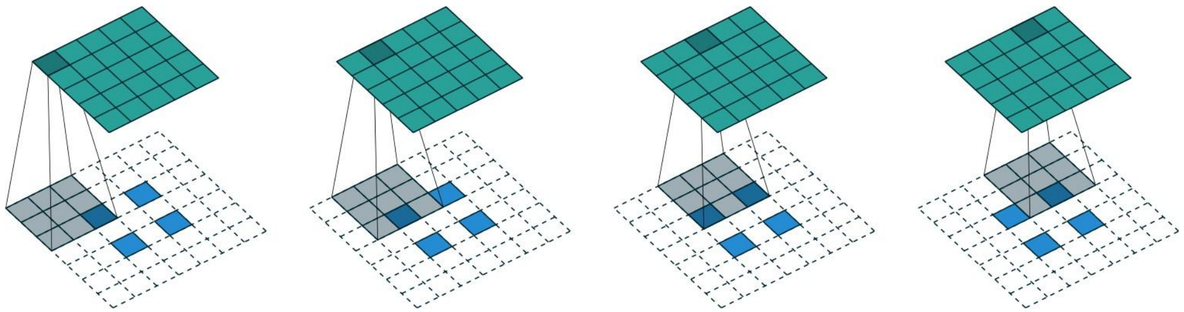
$$o' = s(i' - 1) + k$$



Non-Unit Stride

- No Zero Padding s > 1 and p = 0

$$o' = s(i' - 1) + k$$



- Zero Padding

$$o' = s(i' - 1) + k - 2p$$

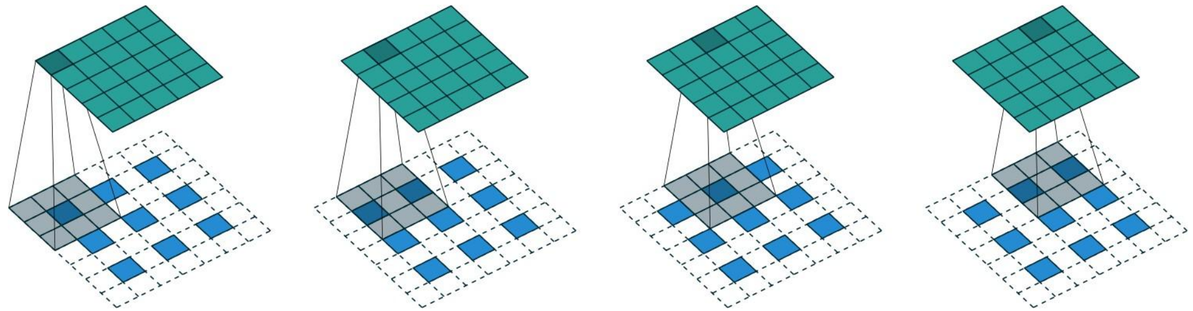


Image Noise

There are two types of noise:

We set f(x, y) as the image and n(x, y) the noise:

- Additive $g(x, y) = f(x, y) + n(x, y)$
- Multiplicative $g(x, y) = f(x, y) * n(x, y)$

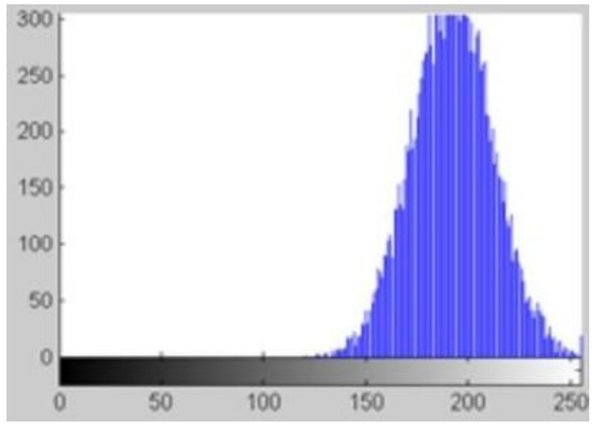
Additive noise is easier to remove using linear filtering or frequency filtering. To remove multiplicative noise non linear filters are needed

Additive Noise - Gaussian

Principal sources of Gaussian noise in digital images arise during acquisition e.g. sensor noise caused by poor illumination and/or high temperature, and/or transmission

e.g. electronic circuit noise. *p* is the probability density function, is the standard deviation of the gray levels, and is the mean of the gray levels.

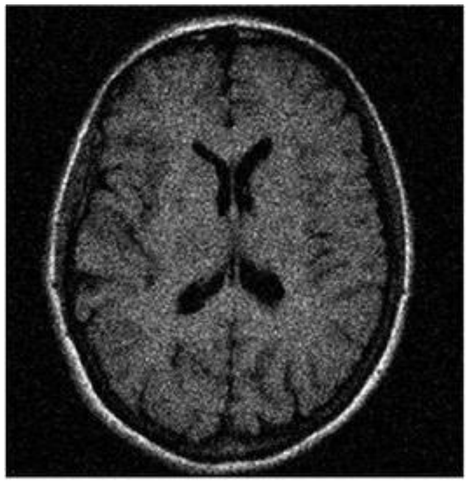
$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$



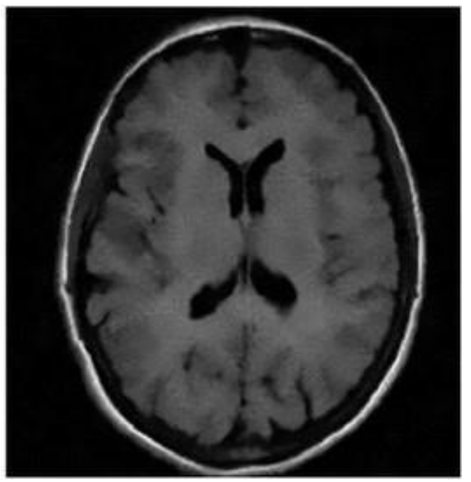
Additive Noise - Gaussian Filter

It smooth the image and is used for noise gaussian noise mitigation. In the spatial domain image smoothing is accomplished by considering a pixel and its neighbors and eliminating any extreme values with a normal kernel

$$\sigma = 1 \qquad X = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

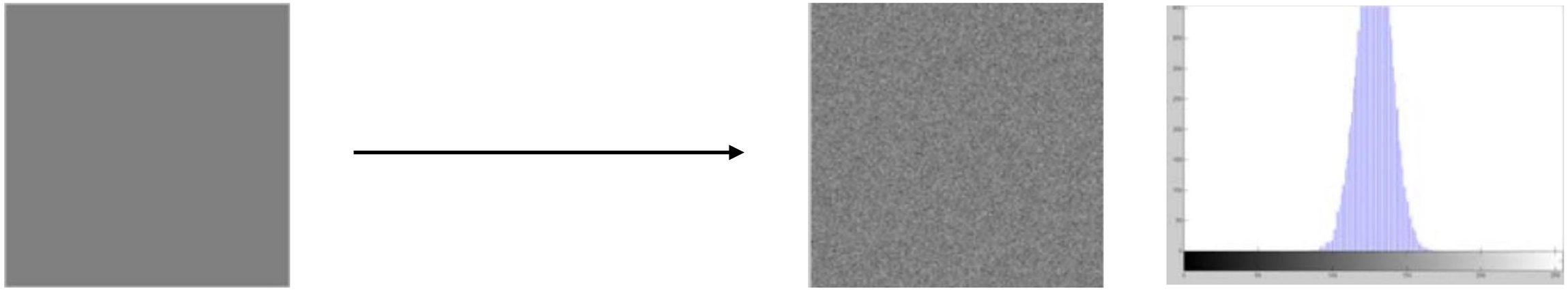


$$\frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \qquad \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



Additive Noise - Poisson

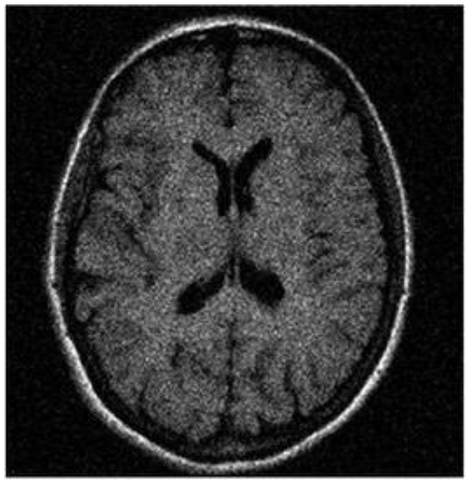
Poisson noise is a signal-dependent noise that can be seen on photon images, and is also called quantum noise. The photon image is an image formed by observation of photons counted in PET and SPECT



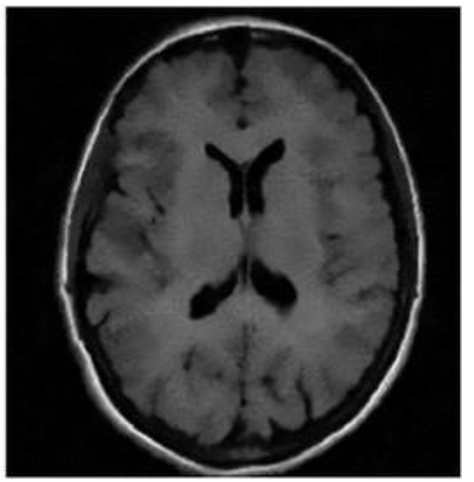
Additive Noise - Mean Filter

It blurs the image more than the Gaussian filter, increasing kernel dimensions produce a greater smoothing

$$\frac{1}{R * C}\begin{bmatrix} l & l & l \\ l & l & l \\ l & l & l \end{bmatrix} \qquad R = 3 \qquad C = 3$$



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Gaussian vs Mean Filter

Original

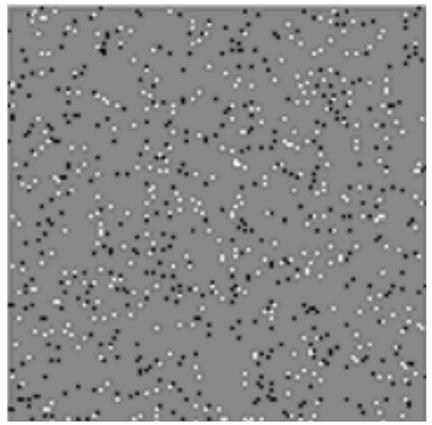
Gaussian

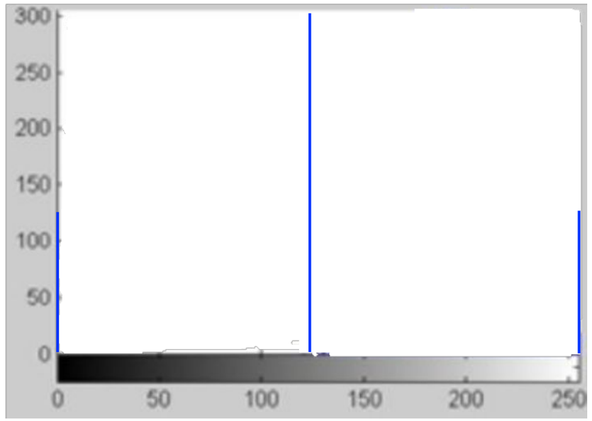
Mean

Salt and Pepper

Salt and pepper noise is an impulse type of noise in images. This noise is generally caused by errors in data transmission, failure in memory cell or analog-to-digital converter errors. It takes the form of randomly occurring white and black pixels, which can significantly deteriorate the quality of an image.

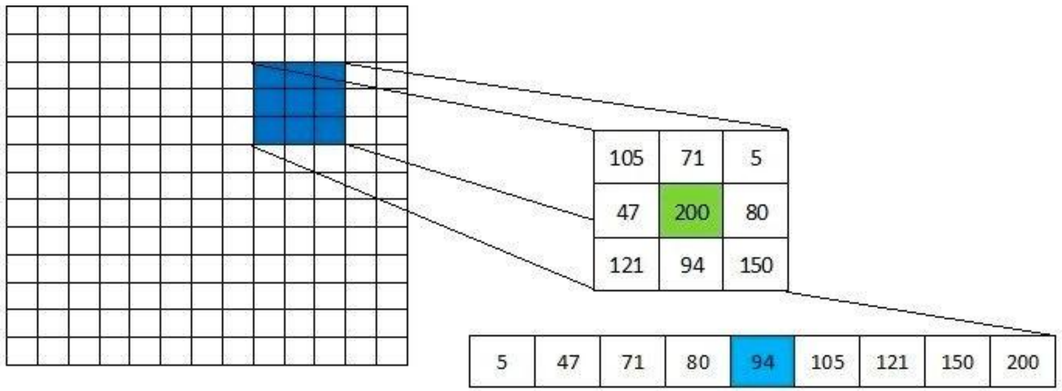
is not additive nor multiplicative. It is obtained
by substituting d/2% (between 0 and 1) of the image pixels with pixels 0 (pepper) and d/2% with pixels 1 (salt).





Median Filter

It is a non linear filter that sorts the pixel values in a small neighborhood and replaces the center pixel with the middle value in the sorted list. The output image needs to be written to a separate image (a buffer), so that results are not corrupted. Neighborhood can be of any size but 3x3, 5x5 and 7x7 are typical. Median filters create a smoothing effect, and large mask sizes create a painted (and blurred) look. As the size of the kernel increases it is more computationally efficient to use an approximation (pseudomedian filter).



Multiplicative Noise - Example

Speckle, speckle pattern, or speckle noise is a granular noise texture degrading the quality as a consequence of interference among wavefronts in coherent imaging systems, such as radar, synthetic aperture radar (SAR), medical ultrasound and optical coherence tomography. Speckle is not external noise; rather, it is an inherent fluctuation in diffuse reflections, because the scatterers are not identical for each cell, and the coherent illumination wave is highly sensitive to small variations in phase changes.

Multiplicative Noise - Homomorphic Filter

Digital images are created from optical images, consisting of a lighting component L(r,c) and a reflectance component R(r,c). Homomorphic filtering attempts to enhance R, while filtering out L

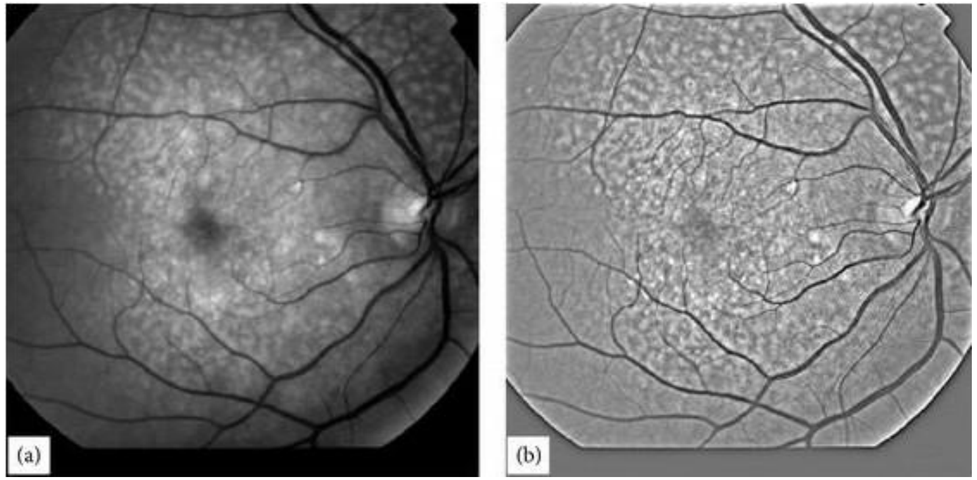
Assumptions:

- Lighting component consists of primarily slow spatial changes and determines overall brightness range
- Reflectance properties of the objects consist of primarily of high spatial frequency information (edges/details) and creates the local contrast

Multiplicative Noise - Homomorphic Filter

The homomorphic filtering process consists of five steps:

1. A natural log transform (base e): it decouples the L(r,c) and R(r,c) components
2. Fourier transform: it puts the image in the frequency domain
3. Filtering
4. Inverse Fourier transform
5. Inverse log function: the exponential, to put back the image in the spatial domain.



Biomedical Computer Vision

October 30th, 2025

Virginia Tasso
virginia.tasso@polimi.it

POLITECNICO MILANO 1863

POLITECNICO MILANO 1863
NECST laboratory