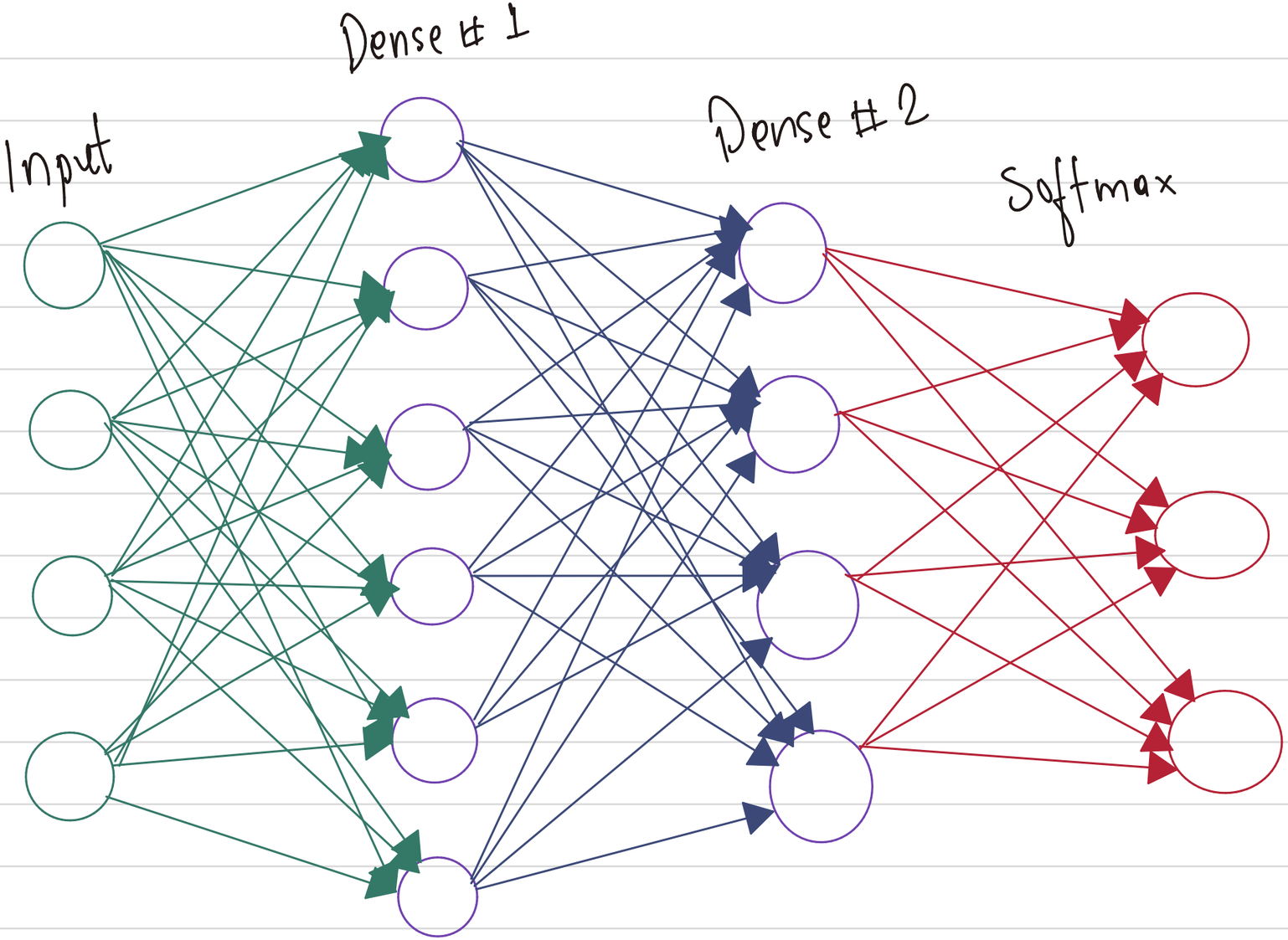
#Day 15, Oct-17, 2024.

#Convolutional Neural Network (CNN):

Definition: CNNs are simply, neural network that uses the Convolutional operation in some or all of their layers.

- Typical Design is that the base layers are mostly CNNs and the final few layers are fully connected layers/ Dense layers.

Fully Connected Network



Important Info.

Input Dimensionality
→ 4

Input to Dense # 1 parameters or the shape of matrix:
4:6
4 ⇒ Input Connected to 6 Hidden layers.

Softmax → used for multi-class (Activation function)

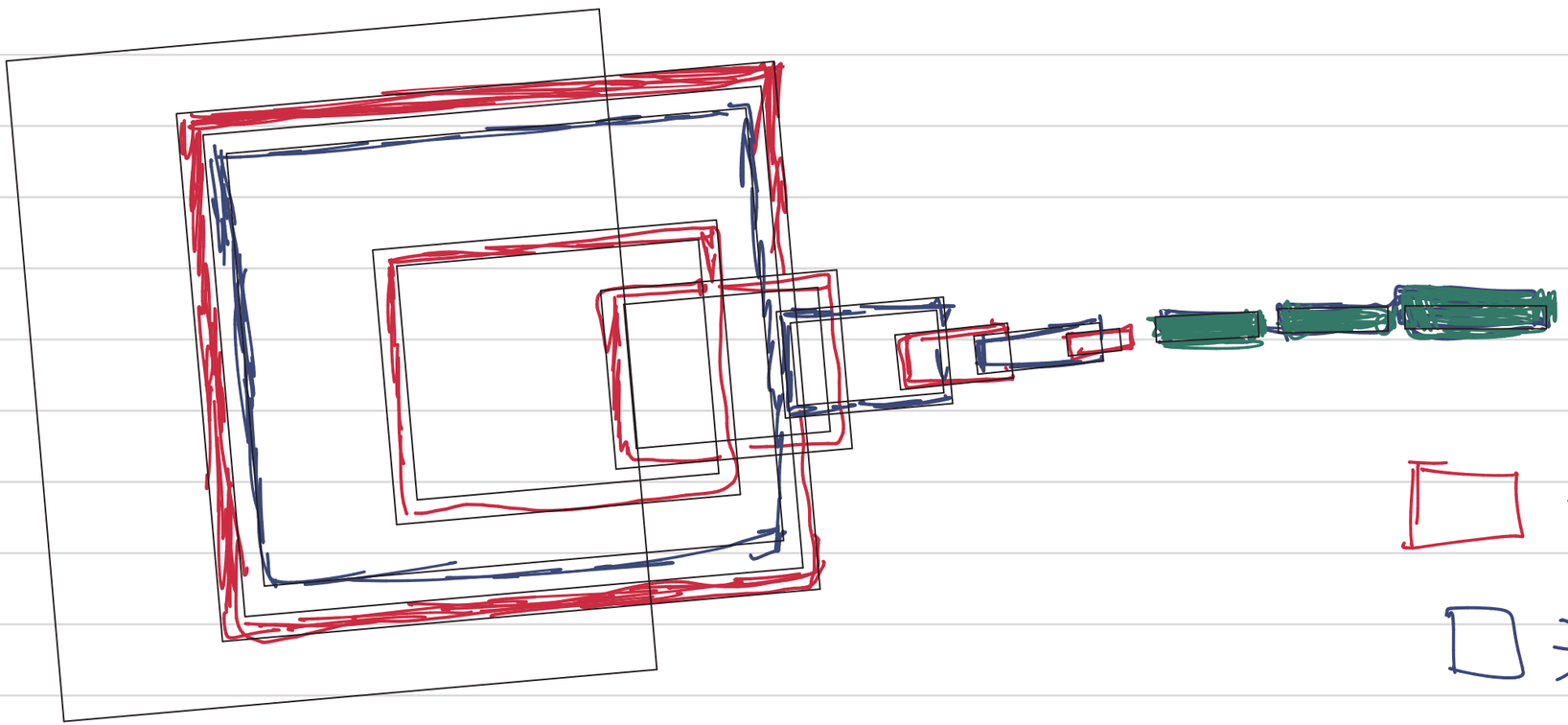
from the fully connected network

4:6 → 6:4 → 4:3

Input Hidden Output

[Projection of Matrix or linear Transformation

4:6 → 6:4 → 4:3]



□ → Max pooling

□ → Convolutional + ReLU

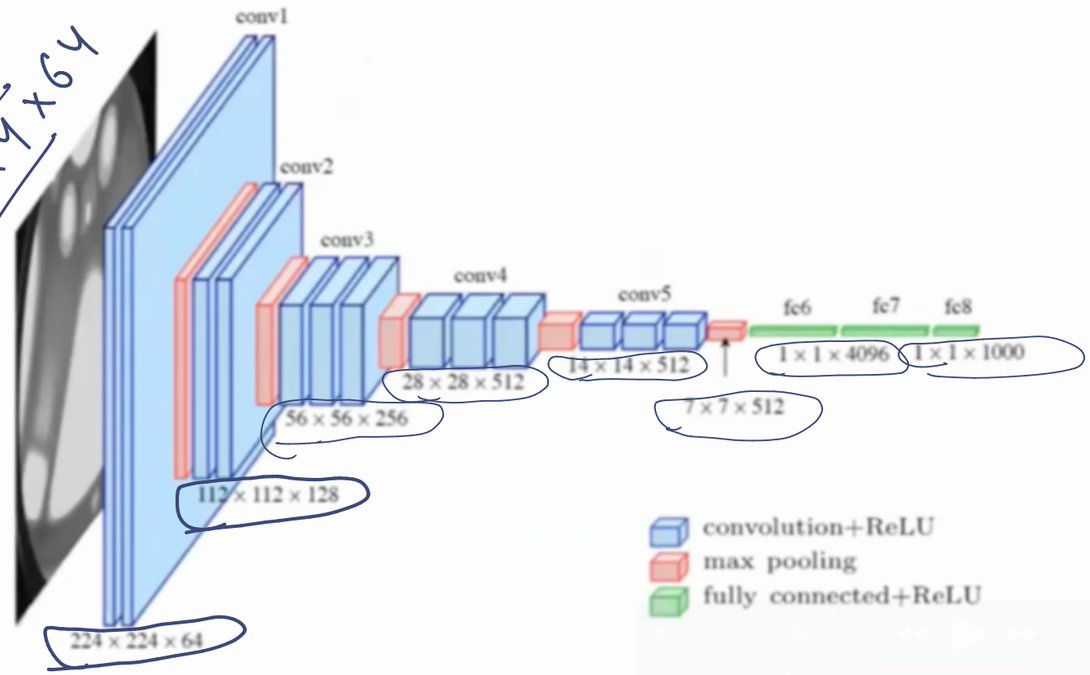
■ → Fully Connected + ReLU.

• Example of classic CNN Architecture (VGG16)

As we go into deeper the height x width of features decreases and the operation is called pooling.

THIRD NEPAL WINTER SCHOOL on AI, 2021
NepAI Applied Mathematics and Informatics Institute for research

Example of a CNN



Example of a classic CNN architecture (VGG16).

nepalschool.naamii.org.np

Src: https://www.researchgate.net/figure/Fig-A1-The-standard-VGG-16-network-architecture

Input: 224x224x64

224 x 224 x 64 → 112 x 112 x 128 → 56 x 56 x 256 → 28 x 28 x 512 → 14 x 14 x 512 → 7 x 7 x 512 | 1 x 1 x 4096 → 1 x 1 x 1000

Features keep on increasing (weight increasing)

#Convolutional Operation: Visual:

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

Kernel / Operator

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

New Image :

6	-9	-8
-3	-2	-3
-3	0	-2

(1) ↓

$7\times1 + 0\times2 + 3\times-1 +$
$4\times1 + 0\times5 + 3\times-1 +$
$3\times1 + 0\times3 + 2\times-1$
$\Rightarrow 7 + 0 - 3 + 4 + 0 - 3$
$+3 + 0 - 2$
$\Rightarrow 8 - 2$
$\Rightarrow 6.$

(3) → $3\times1 + 0\times3 + 8\times-1 +$
$3\times1 + 0\times8 + 4\times-1 +$
$2\times1 + 8\times0 + 4\times-1$
$\Rightarrow 3 + 0 - 8 + 3 + 0 - 4 + 2 + 0 - 4$
$\Rightarrow 3 - 5 - 4 + 2 - 4$
$\Rightarrow -8$

(2) ↓ $2\times1 + 3\times0 + 3\times-1 +$
$5\times1 + 3\times0 + 8\times-1 +$
$3\times1 + 2\times0 + 8\times-1$
$\Rightarrow 2 + 0 - 3 + 5 + 0 - 8 + 3 + 0 - 8$
$\Rightarrow -9$

Similarly Repeat steps 1, 2 and 3 for all cells to be filled with new pixels value.

CNN's

(1) It is a local operation

(2) the same filter is applied at all locations

(3) Different filters (Kernels) with different weights, respond to different attributes of the signal.

Cross-Correlation is used in CNN because learning of the appropriate parameters.

Flip one of the Signals in Convolutional but not required in CNN.

So, the convolution is actually Cross-Correlation in the Convolutional Neural Network (CNNs).

(1) Convolutional NN are local or Sparse operators because if we observe what we are doing then-

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

$*$

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

$\Rightarrow$

6	-9	-8
-3	-2	-3
-3	0	-2

the value 6 is obtained not from the entire 6x6 Matrix but 3x3 local matrix which is a subset of 6x6 Matrix.

#So, CNN's view the output 6 is only connected to 3x3 Matrix parameters

7	2	3
4	5	3
3	3	2

$*$

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

$\Rightarrow$ 6 output:

About Kernel

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

if the 6x6 is whole Dense Connection we would use whole Matrix with 6x6 Kernel.

Image Response to the Kernel, we can use different Kernels and observe how would Image response to those Kernels.

What does Kernel represent itself?

→ Detector of Vertical Edges, the Kernel itself is a Vertical Edge Detector or the edge detection.

→ Kernel can be bluring or used as bluring.

→ In Signal processing, Kernel will do Smoothing. And Edge Detector responds to the edges.

So, the Kernel is the main spice in the Convolutional ANN.

Network learn these filters instead of explicitly providing the Kernel filters like in classical Signal & Digital Image Processing.

Motivation for CNN.

- As neural networks became bigger, usually fully connected networks started becoming untenable due to its large parameter size.
  - CNNs use significantly less parameters for comparable performance.

- CNNs use sparse local connections - its a form of implicit regularization.
- using pooling with CNNs gives translational invariance.

Computational Efficiency for larger & complex Neural Networks.

CNN's have less parameters which is good for training.

Implicit Regularization:

L1 → Reduces absolute value to zero's (In L1 only important ones are values '1') weights are zero, Sparse parameters

L2 → Keep the features, doesn't Reduces the weights to not absolute zero's like L1

Over Complete problems - Large possibility of Solutions for a given problem, hard to choose one Solution.

Making CNN is like already making strong Regularized Model.

Why load or Sparse is necessary!

→ First layer learns edges and Smooth

→ Second layer responsive to curvature, convex or concave shapes or geometric shapes

→ Deeper Next Layer CNN responsive to structures & class Structures/ objects like eyes, nose, ears.

So, the CNN's is bottom-up approach we start from local to global and from edges to objects detection/ Simple to Complex.

pooling reduces the image sizes as shown in above operation

EKG Signal are generated by devices hearing heart-beat.

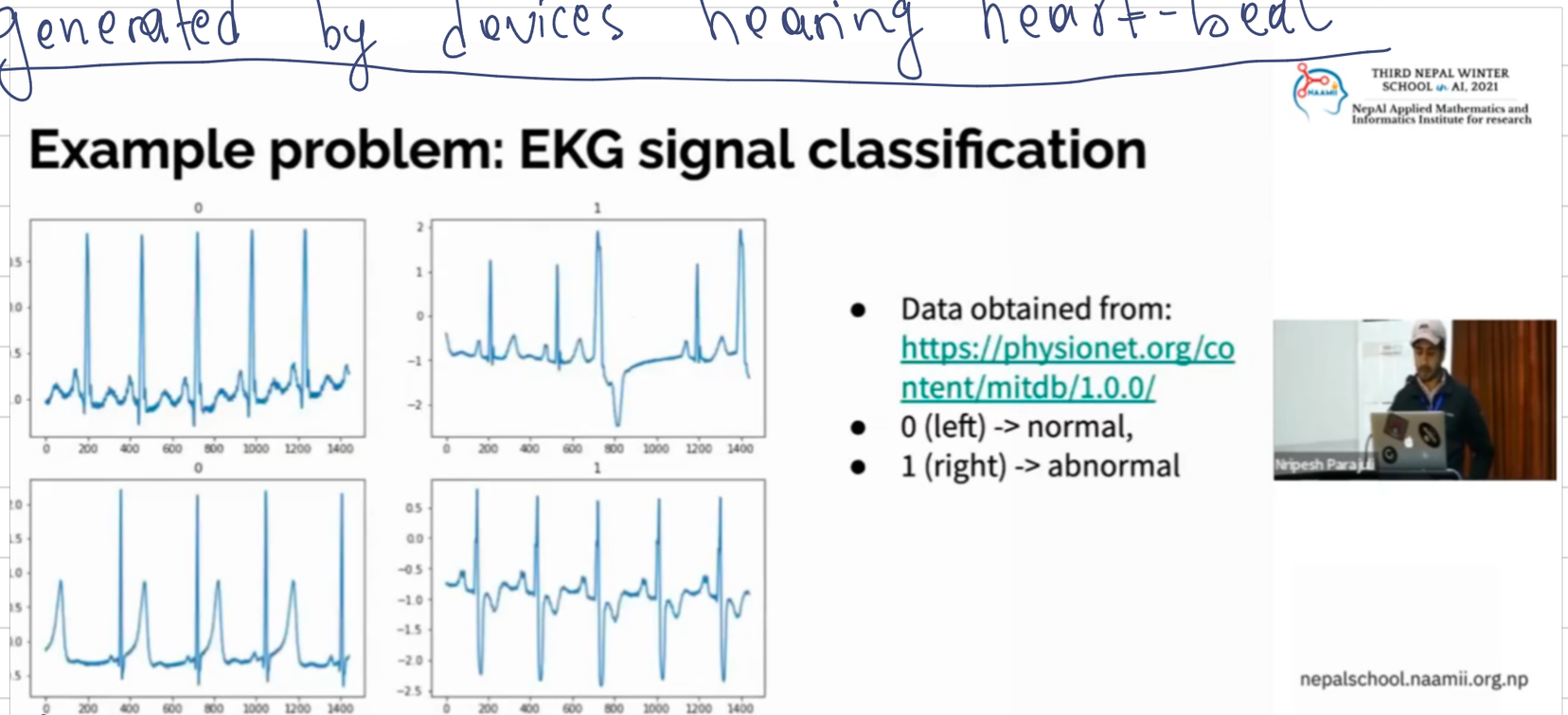
Classifier

0 (left) → Normal

and

1 (right) → abnormal.

→ Normal vs abnormal Signal.



Edge Detection:

Edges are straight lines or curves in the image plane across which there is a significant change in image brightness.

Edges correspond to locations in images where the brightness undergoes a sharp change, so a naive idea would be to differentiate the images whose the brightness undergoes a sharp change, of the derivative $I'(x)$ is large.

Lets take a image at $x=50$ a peek value but also subsidiary peaks at other locations (eg: $x=75$) which is noise.

→ Smoothing the Image Noise $x = 75$ gets vanished.

Fig 1:

Intensity profile of $I(x)$ along one-dimensional section across edge at $x = 50$

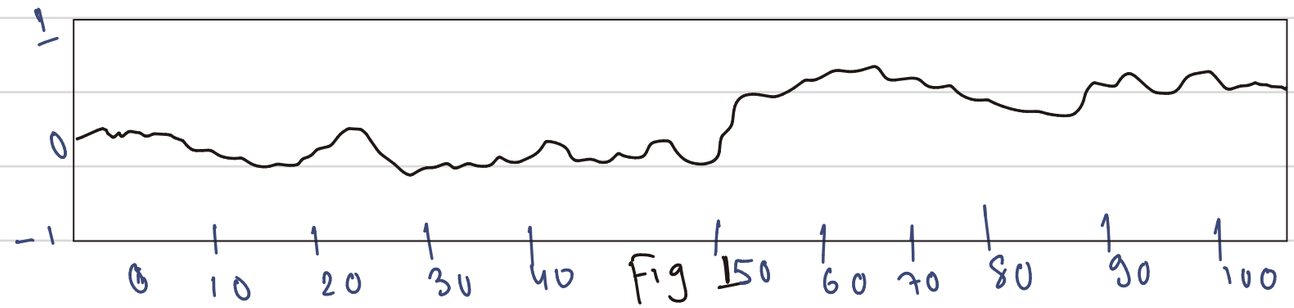


Fig 2:

Derivative of Intensity $I'(x)$ where large value correspond to edges, but the function is noisy.

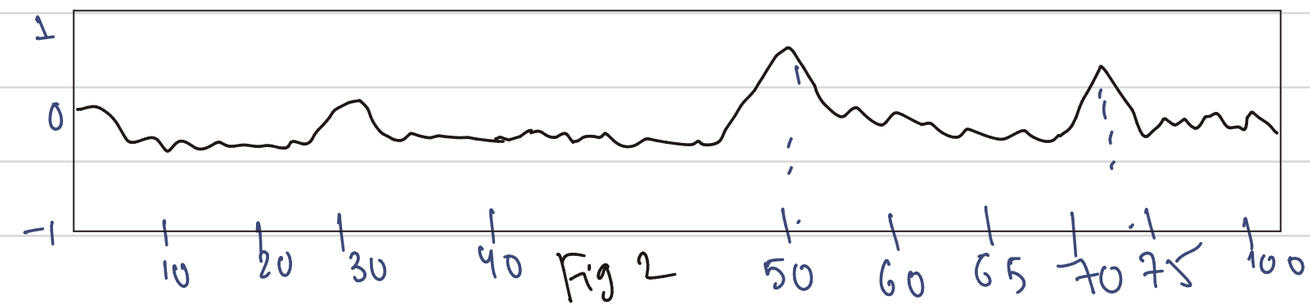
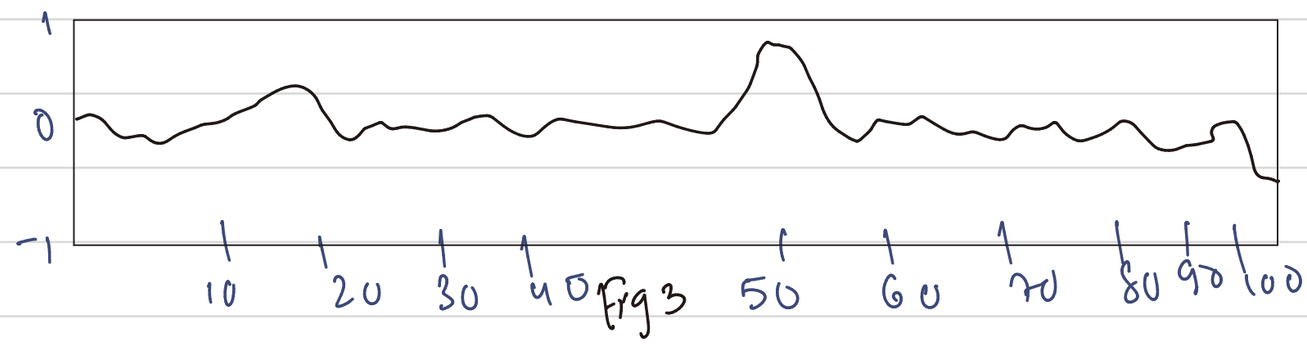


Fig 3: Derivative of Smoothed version of the Intensity $(I \times G_6)'$ that can be computed in one singlestep as Convolutional & noise vanished.



How to Smooth Images?

Answer is using Gaussian filter that does the weighted average that weights the nearest pixels the most, then gradually decreases the weight for more distant pixels.

We have modeled noise using Gaussian probability distribution.

And the Gaussian function is

$$N_\sigma(x) \Rightarrow \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2}$$

$\sigma \Rightarrow$ Standard Deviation

$x \Rightarrow 0$ (mean)

$\rightarrow$ 1-D

$$N_\sigma(x,y) \Rightarrow \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2-y^2}{2\sigma^2}}$$

in 2-D.