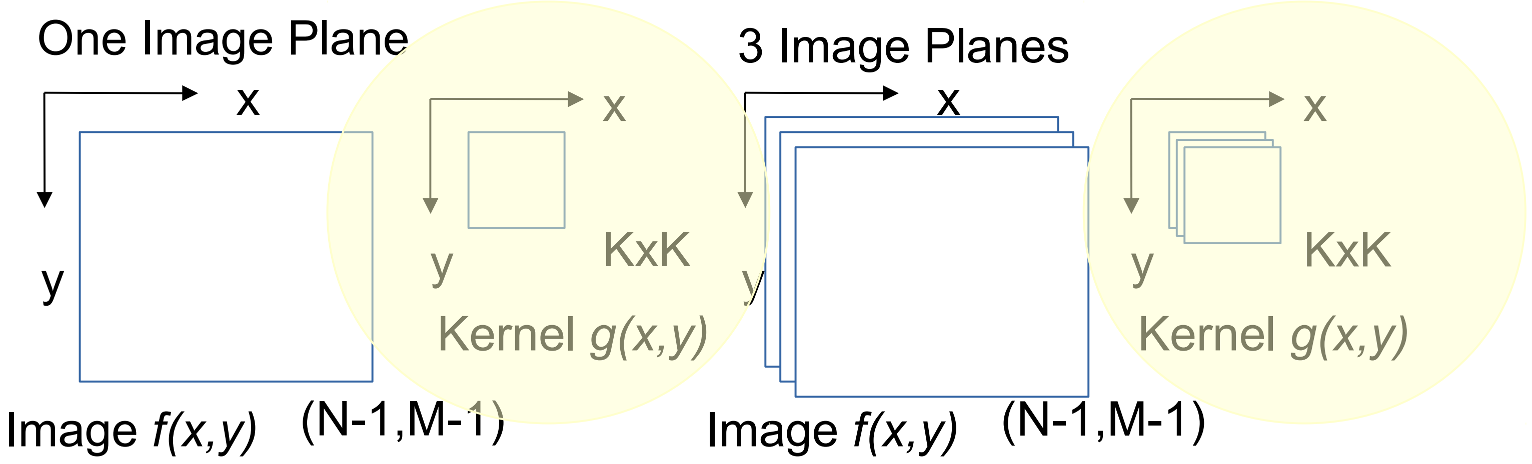
Kernel Design for Convolution and Convolutional Neural Networks CNN

Example: Given the following two convolutions, one is with image depth = 1, kernel depth = 1, and the other is with image depth (layer) = 3, and kernel layers = 3.



Now, design methodology can be classified in 2 categories:

① 1. Derived Kernel: Its design is based on mathematical formulation, which is limited to what we know in theory, and

② 2. Learned Kernel: its design is based on learning via neural networks, which is based on learning but often poses challenges of having lack of fully explanation and understanding of how and why the kernel is like the way it is after the training.

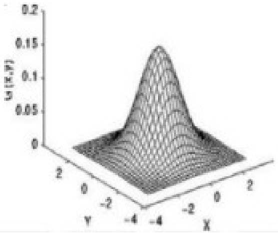
Harry Li, Ph.D. 2017

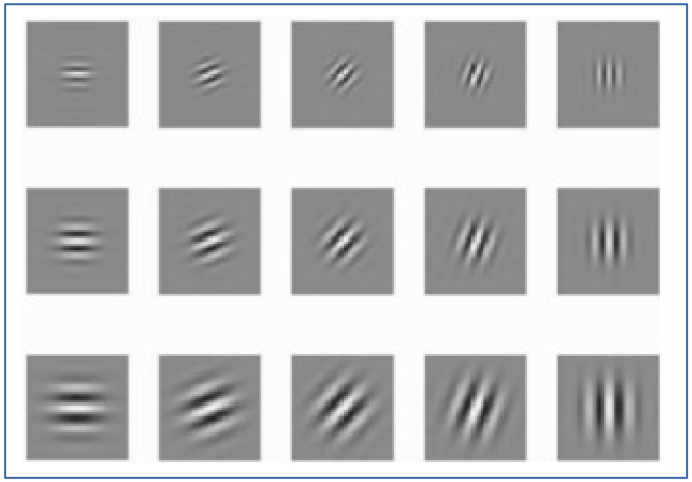
Comparison of Kernel Design Methodology

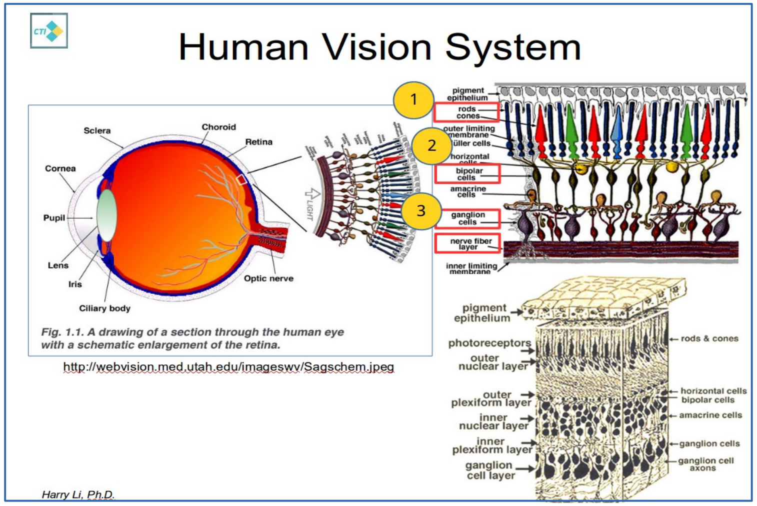
Derived Kernel

1

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma)=\exp\left(-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}\right)\exp\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right)$$







Human Vision System

Fig. 1.1. A drawing of a section through the human eye with a schematic enlargement of the retina.

http://webvision.med.utah.edu/imageswv/Sagschem.jpeg

Harry Li, Ph.D.

2

Learned Kernel

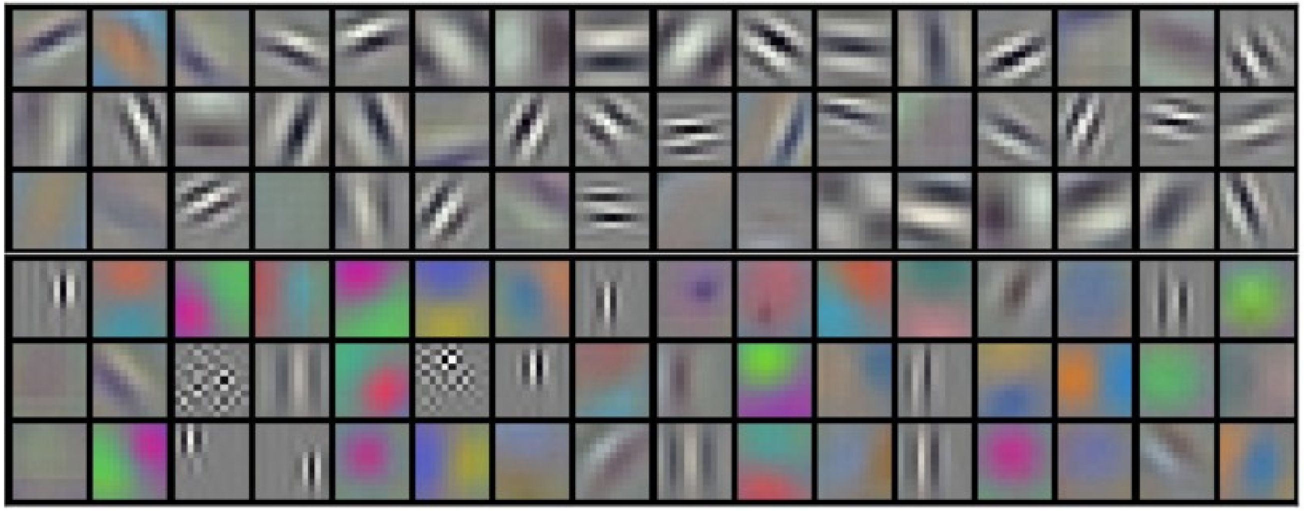


Figure 3: 96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2.

Ref:

ImageNet Classification with Deep Convolutional Neural Networks, pp. 6.

Kernel Design for Derived Kernels

Example:

1) Given a digital image $f(x,y)$, design 4 edge detectors to pick up vertical edge components.

Sol

First, use forward difference technique,

$\longrightarrow y$ $\quad f(x,y)$

$x \downarrow$

0	100	100
0	100	100
0	100	100

$$\frac{\partial f(x,y)}{\partial y} \simeq f(x,y+1) - f(x,y)$$

$\longrightarrow y$

-1	+1

$x \downarrow$

forwards difference

Backwards difference

$$\frac{\partial}{\partial y} f(x,y) \simeq f(x,y) - f(x,y-1)$$

$\longrightarrow y$

$x \downarrow$

-1	+1

Central difference

$$\frac{\partial}{\partial y} f(x,y) = \frac{1}{2}\left[\left.\frac{\partial f(x,y)}{\partial y}\right|_{\text{Forward}} + \left.\frac{\partial f(x,y)}{\partial y}\right|_{\text{Back}}\right]$$

$$= \frac{1}{2}[f(x,y+1) - f(x,y) + f(x,y) - f(x,y-1)]$$

$$= \frac{1}{2}[f(x,y+1) - f(x,y-1)] \cdots (3)$$

$\longrightarrow y$

$\frac{1}{2}$ $x \downarrow$

-1		+1

Derive LoG Kernel

Example: LoG stands for Laplace of Gaussian

First, Laplace operator is given as:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad ...(1)$$

First, 1D Gaussian function

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

Then, 2D Gaussian function,

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}} \quad ...(5)$$

Assume $\mu_x = \mu_y = 0$

$$\frac{\partial}{\partial x} G(x,y) = -\frac{x}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \ldots (3)$$

Hence

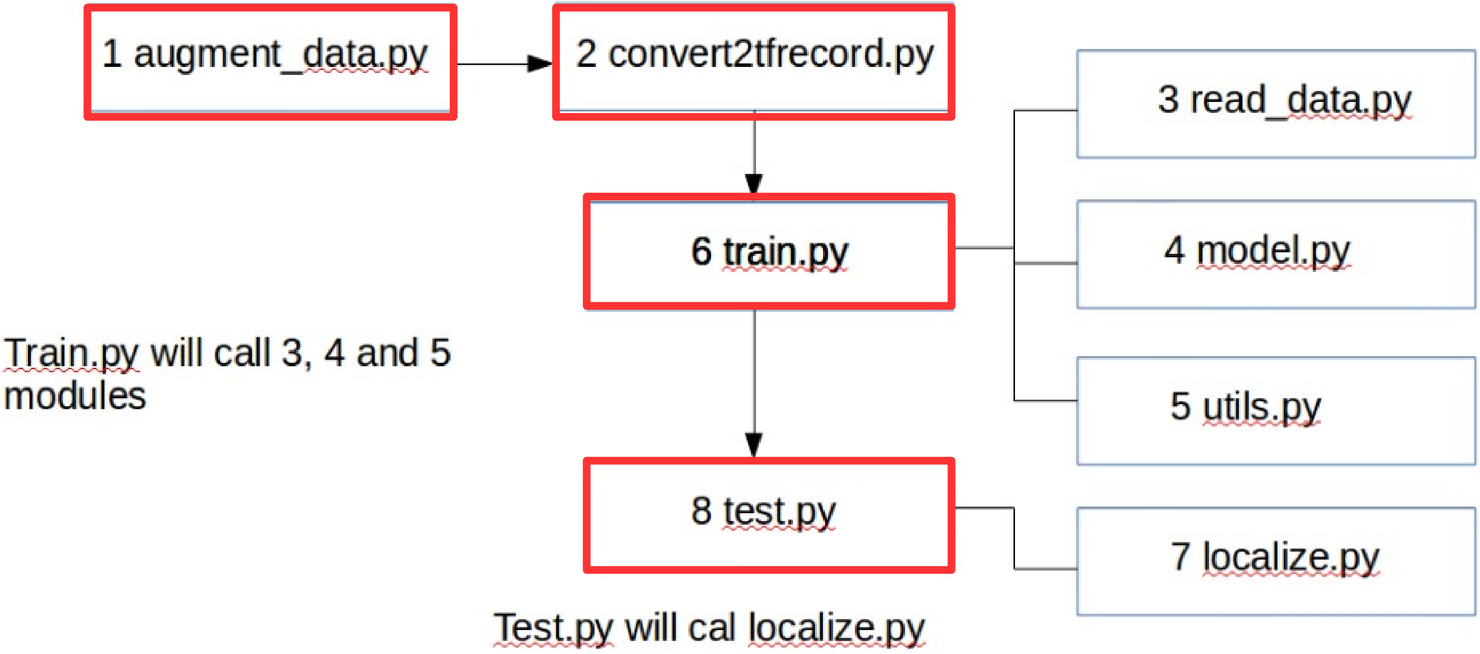
$$\frac{\partial^2}{\partial x^2} G(x,y) = -\frac{1}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{x^2}{\sqrt{2\pi}\sigma^5} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \ldots (4)$$

$$\frac{\partial^2}{\partial y^2} G(x,y) = -\frac{1}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{y^2}{\sqrt{2\pi}\sigma^5} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \ldots (5)$$

$$\nabla^2 G(x,y) = \frac{x^2+y^2-2\sigma^2}{\sqrt{2\pi}\sigma^5} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

CTI One Sample Code

Deep Learning Modules



Train.py will call 3, 4 and 5 modules

Test.py will cal localize.py

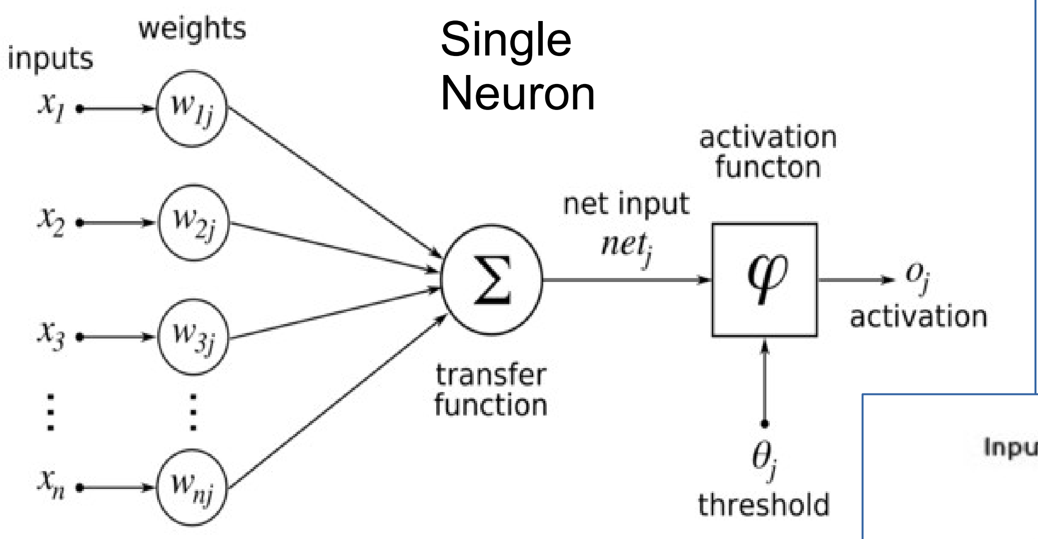
Preprocessing for Deep Learning

augment.py

```
# Augment the image dataset with rotation and blurring
 for f in file_names:
     img = cv2.imread(f)
     if img is not None:
         print("Processing" + f)
         M = cv2.getRotationMatrix2D((img.shape[1] / 2, img.shape[0] / 2),
                                      10, 1)  # rotation matrix by 10 degree
         rotate1 = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))
                                             # rotate image and assign it back
         M = cv2.getRotationMatrix2D((img.shape[1] / 2, img.shape[0] / 2),
                                      -10, 1) # rotation matrix counterwise
         rotate2 = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))
                                             # rotate image and assign it back

         blur1 = cv2.GaussianBlur(img, (5, 5), 3) # 5 by 5 kernel, sigma 3
         blur2 = cv2.GaussianBlur(img, (7, 7), 5)
         blur3 = cv2.GaussianBlur(img, (9, 9), 7)
```

Feed Forward Neural Networks



Single Neuron

https://d4datascience.wordpress.com/2016/09/29/fbf/

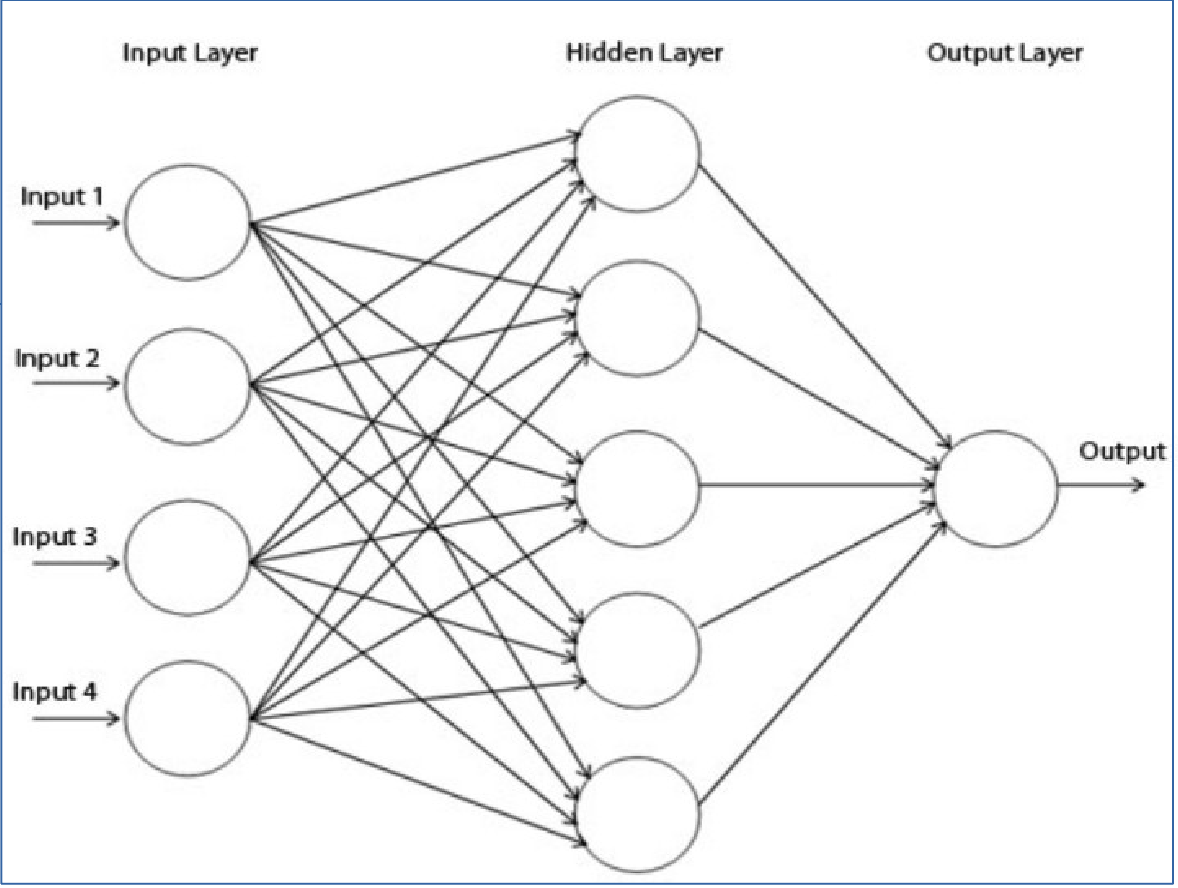
$$d(\vec{x}) = \vec{w}^t \vec{x} \quad \ldots (1)$$

Assume phi = 0

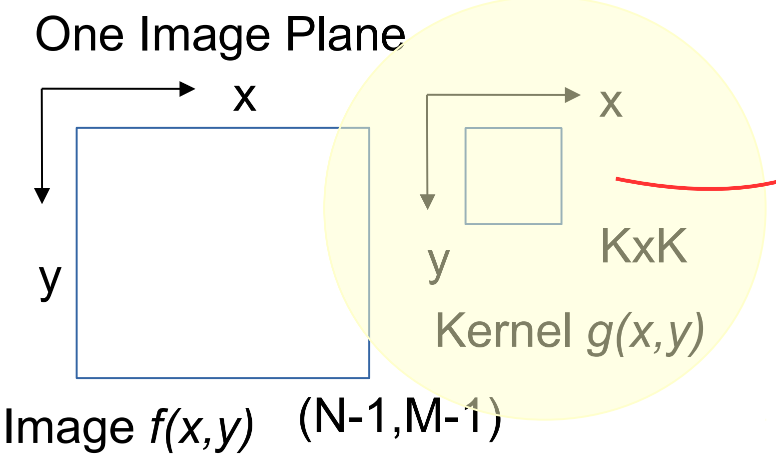
where $\vec{w}^t = (w_1, w_2, \cdots, w_{n+1})$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{pmatrix}$$

Multi-layer feed forward neural networks



Kernel Coefficients to Neural Nets



One Image Plane

x

y

Image *f(x,y)*   (N-1,M-1)

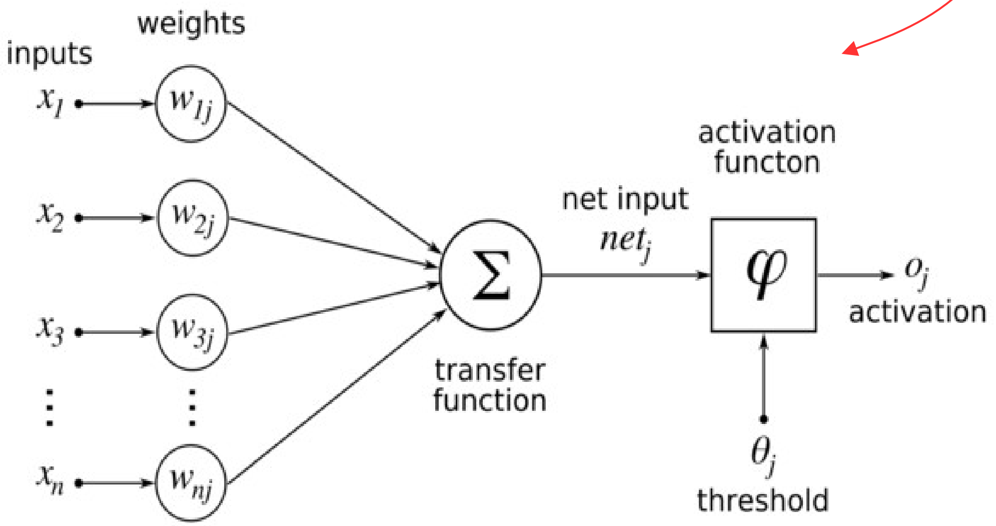
x

y

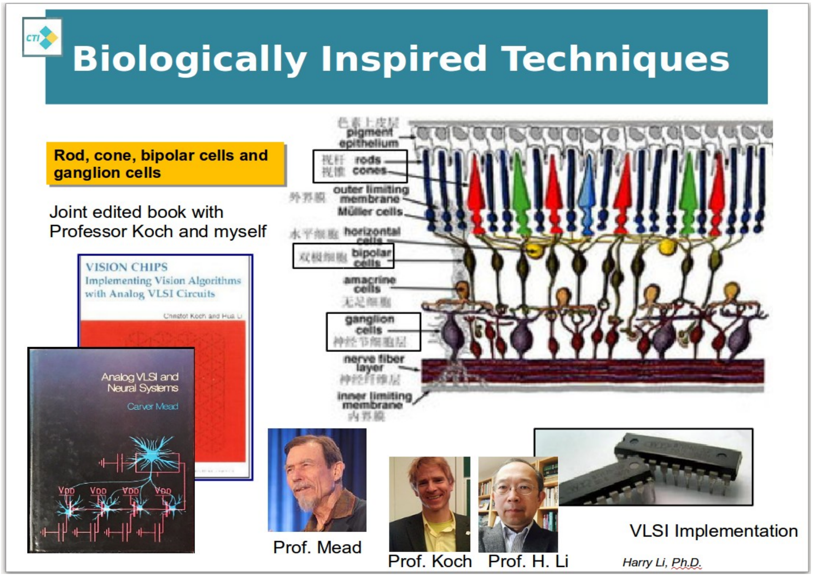
KxK

Kernel *g(x,y)*

w_11	w_12	w_13
w_21	w_22	w_23
w_31	w_32	w_33



Neural Nets: Biological Inspirations



Biologically Inspired Techniques

Rod, cone, bipolar cells and ganglion cells

Joint edited book with
Professor Koch and myself

Prof. Mead

Prof. Koch   Prof. H. Li

VLSI Implementation

Harry Li, Ph.D.

Harry Li, Ph.D.