NUS AI SUMMER EXPERIENCE 2019

Computer Vision and Deep Learning

Li Xinke

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

01 Image Formation

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

Image Formation – solid state camera

– Separate photodiode located at each location.

– Photodiodes are arranged in either a linear array or a rectangular array.

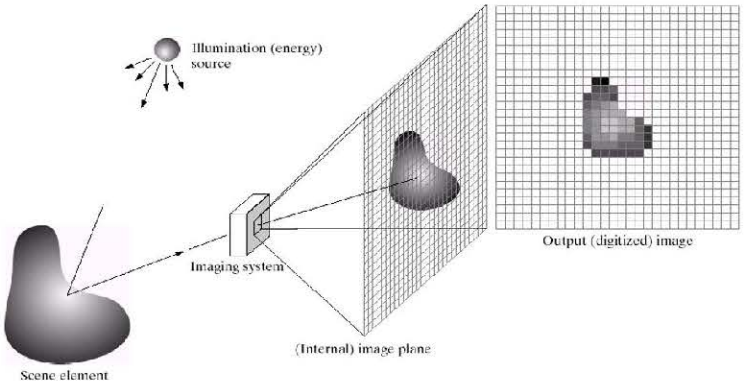
– Resistivity of the photodiode decreases proportionally to the amount of light falling on it.

– The output signal voltage is dependent on the resistance of the photo-conductor.

– Output video is obtained by scanning each diode in ordered sequence to produce a series of voltage pulses representing the pixel value at the respective location.

– Most common type of solid-state camera : CCD (or charge-coupled device)

Image Formation



a b c d e

FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Image Formation – attributes

Image attributes

– An image is a 2D light-intensity function denoted by f(x,y).

– The value or amplitude of f at spatial coordinates (x,y) gives the intensity (brightness) of the image at that point.

– f(x,y) is non-negative and finite.

– f(x,y) must be digitized both spatially and in amplitude for computer processing.

– Digitization of spatial coordinates (x,y) is called spatial quantization or discretization or sampling.

– Amplitude digitization is called gray-level quantization.

Image Formation - attributes

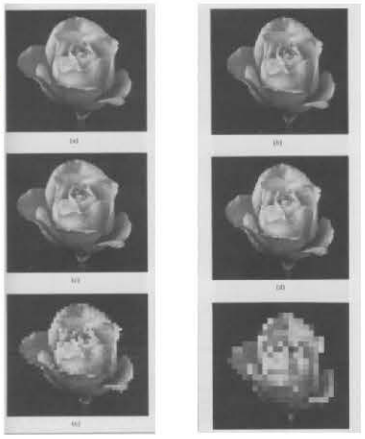
- Image attributes – The resolution of the image refers to the N x M array to which an image is sampled.

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & & & \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

- – Common image resolution has the following characteristics: N=2^n; M=2^k, where n and k are integers.
- – gray level G=2^m, where m is the no. of bits.

Image Formation - resolution

- Examples (spatial resolution with a fixed number of grey levels)



Grey level 0 – Grey
Grey level 255 – White

512x512

128x128

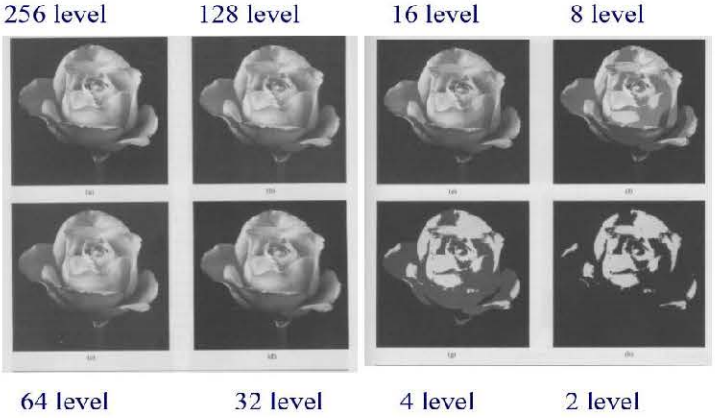
Image size is fixed
Pixel size is increasing.

Instead, we can keep pixel size fixed while down sizing images!

32x32

Image Formation - resolution

- Examples (gray level resolution with a fixed spatial resolution)



02 Preprocessing of Images

Preprocessing of Images

Image Enhancement

– spatial domain methods:

- pixel by pixel transformations (point processing).
- neighborhood operations (area processing).
- algorithms operate directly on pixel values in an image in spatial domain.

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

Preprocessing of Images

– spatial frequency domain methods:

• image filtering in spatial frequency domain

• image must be transformed from spatial domain to spatial frequency domain by Fourier transform before filtering

• Filters can be designed in either spatial or spatial frequency domains, but used only in spatial frequency domain.

• Inverse Fourier transform is needed to convert the results back to spatial domain.

Preprocessing of Images – point processing

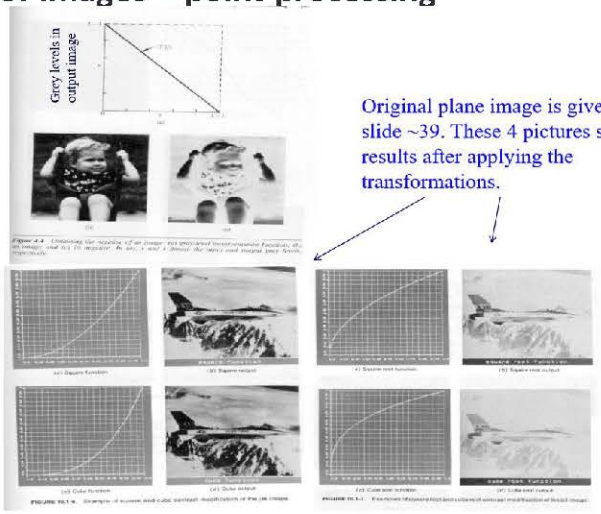
• Point processing (N=1):

– T becomes a gray level transformation function

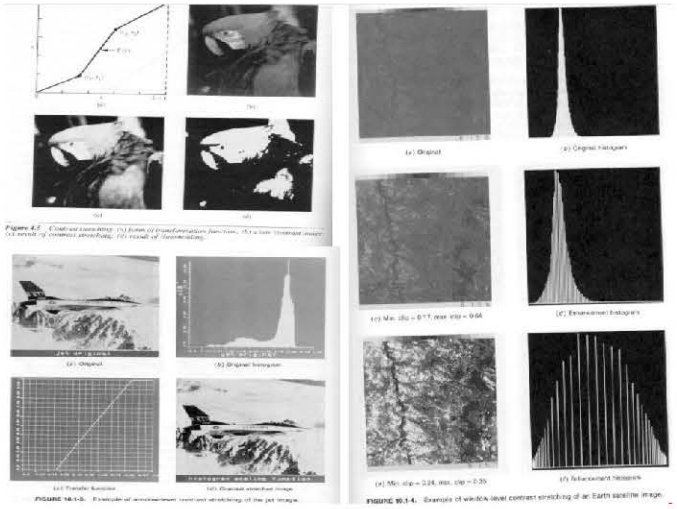
where r, s are variables denoting the gray level of f(x,y) and g(x,y) at (x,y).

– Look-up-table (LUT) can be used to implement the transformation function.
– By defining the LUT, point processing can be performed by indexing the LUT.

Preprocessing of Images – point processing



Preprocessing of Images – point processing (histogram stretching)



Preprocessing of Images – image filtering

- Image filtering can be performed in spatial domain or in spatial frequency domain.
- A filter with desired spatial frequency response is designed to enhance the corresponding spatial frequency components in the image.
- The filter can be designed in the spatial domain or the spatial frequency domain.
- Image filtering in spatial domain is achieved by convolution.
- Image filtering in spatial frequency domain is achieved by spectral multiplication.

Preprocessing of Images – image filtering

Fourier Transform (FT) (Based on G & W)

- The two dimensional FT and inverse FT are:

$$F(u,v) = \Im\{f(x,y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp[-j2\pi(ux+vy)]dxdy$$

$$f(x,y) = \Im^{-1}\{F(u,v)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)\exp[j2\pi(ux+vy)]dudv$$

Preprocessing of Images – image filtering

- u and v are frequency variables.
- The magnitude, phase and power of the spatial frequency spectrum are

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2}$$
$$\phi(u,v) = \tan^{-1}[I(u,v)/R(u,v)]$$
$$P(u,v) = |F(u,v)|^2$$

where R(u,v) is the real part and I(u,v) is the imaginary part.

Preprocessing of Images – image filtering

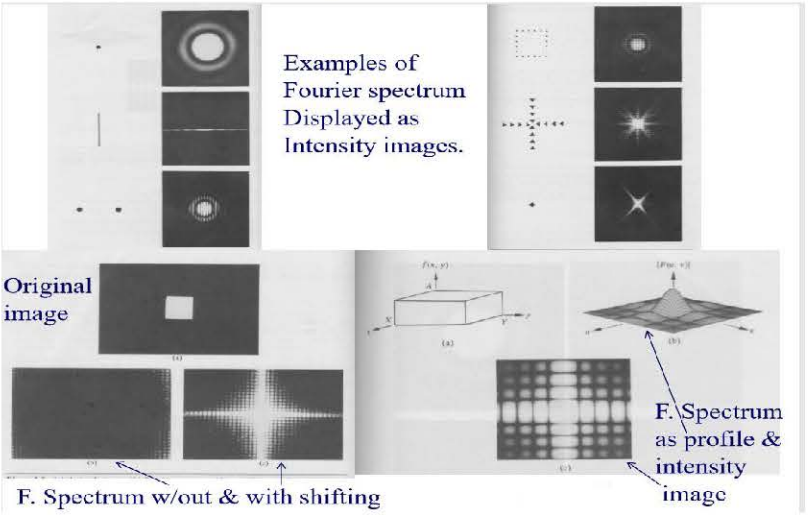
- Discrete Fourier Transform (DFT)
- • The two dimensional DFT and inverse DFT are:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp[-j2\pi(ux/M + vy/N)]$$

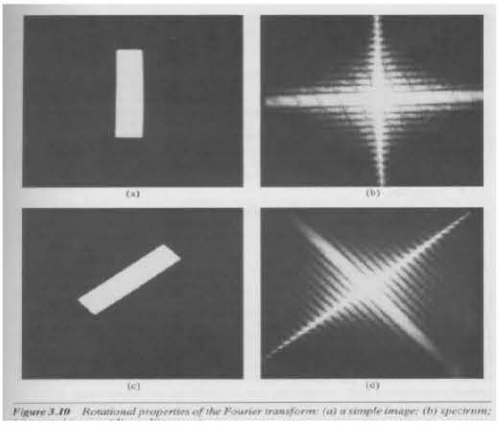
$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp[j2\pi(ux/M + vy/N)]$$

- where N and M are the dimensions of the 2D matrix

Preprocessing of Images – image filtering



Preprocessing of Images – image filtering



Rotational property of FT

Preprocessing of Images – image filtering

Convolution and Correlation

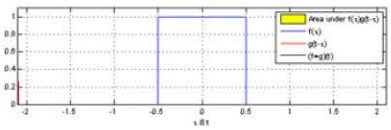
- The 1D convolution is :

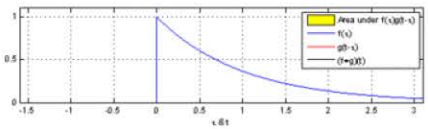
$$f(x) \otimes g(x) = \int_{-\infty}^{\infty} f(\alpha) g(x-\alpha) d\alpha$$

- Convolution in time / spatial domain is equivalent to multiplication in frequency domain and vice versa.

$$f(x) \otimes g(x) \Leftrightarrow F(u)G(u)$$
$$f(x)g(x) \Leftrightarrow F(u) \otimes G(u)$$

Preprocessing of Images – image filtering





Preprocessing of Images – image filtering

Convolution and Correlation

- The 2D convolution is :

$$f(x,y) \otimes g(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha,\beta) g(x-\alpha, y-\beta) d\alpha d\beta$$

- Convolution in spatial domain is equivalent to multiplication in spatial frequency domain and vice versa.

$$f(x,y) \otimes g(x,y) \Leftrightarrow F(u,v)G(u,v)$$
$$f(x,y)g(x,y) \Leftrightarrow F(u,v) \otimes G(u,v)$$

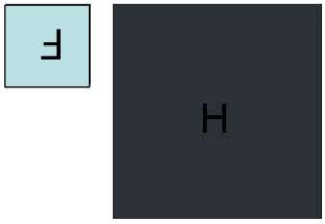
Preprocessing of Images – image filtering

Convolution and Correlation

- Flip the filter in both dimensions (bottom to top, right to left)
- Then apply cross-correlation

$$G = H \star F$$

Notation for convolution operator



Preprocessing of Images – image filtering

Convolution and Correlation

- The 1D correlation is :

$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f^*(\alpha) g(x+\alpha) d\alpha$$

- Correlation in time domain is equivalent to multiplication in frequency domain and vice versa as follows.

$$f(x) \circ g(x) \Leftrightarrow F^*(u) G(u)$$

$$f^*(x) g(x) \Leftrightarrow F(u) \circ G(u)$$

Preprocessing of Images – image filtering

Convolution and Correlation

- The 2D correlation is :

$$f(x,y) \circ g(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f^*(\alpha,\beta) g(x+\alpha, y+\beta) d\alpha d\beta$$

- Correlation in spatial domain is equivalent to multiplication in spatial frequer

$$f(x,y) \circ g(x,y) \Leftrightarrow F^*(u,v) G(u,v)$$

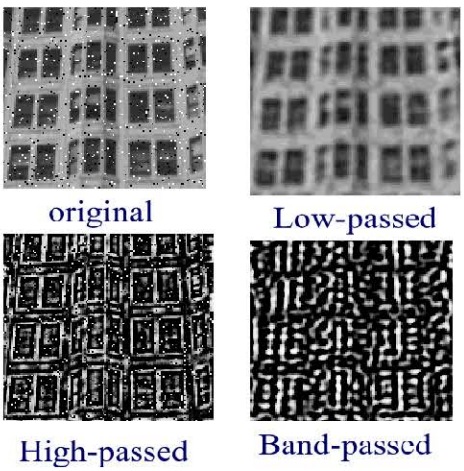
$$f^*(x,y) g(x,y) \Leftrightarrow F(u,v) \circ G(u,v)$$

Preprocessing of Images – image filtering

– Filtering an image involves the following steps:

- Transform image into spatial frequency domain using FT.
- Filter the spatial frequency domain with predesigned filters.
- Transform from spatial frequency domain into image domain using Inverse FT.

Preprocessing of Images – image filtering



Preprocessing of Images – image filtering

Band-pass filtering

– Band-pass filtering enhances edges and suppresses image noise (in frequency domain).

– Equivalent band-pass action in spatial domain (more details later):

• Gaussian + First Derivative Operator :

–First derivative of Gaussian

• Gaussian + Second Derivative Operator :

–Laplacian of Gaussian

Preprocessing of Images – image filtering

Neighborhood Operations (Spatial Domain)

– Most neighborhood operations can be considered as image filtering by convolution with a convolution kernel

– The operation is performed in spatial domain.

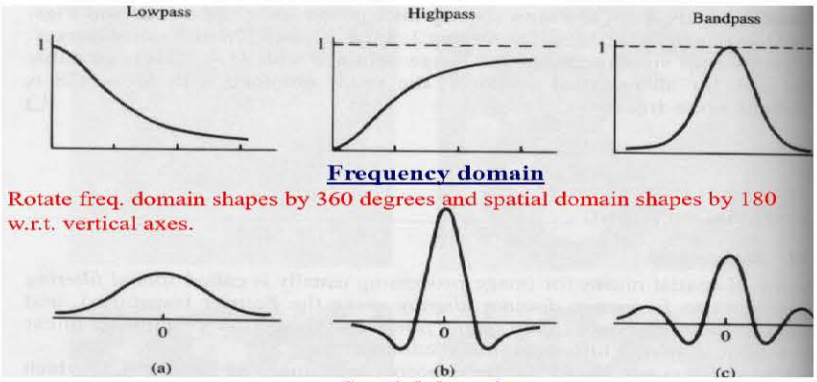
– Low-pass, high-pass and band-pass filters are designed and used in spatial domain.

– Block averaging is considered as a low-pass spatial filter (see the figure at bottom right FT profile – Here f(x,y) is block averaging in continuous domain.)

– Weighted averaging uses a Gaussian kernel.

Preprocessing of Images – image filtering

Equivalent Cross-Sectional shapes in spatial and frequency domains



Circular symmetry frequency domain filters and their spatial domain counter parts. The pictures show cross-sectional profiles.

Preprocessing of Images – image filtering(smoothing)

Gaussian Smoothing

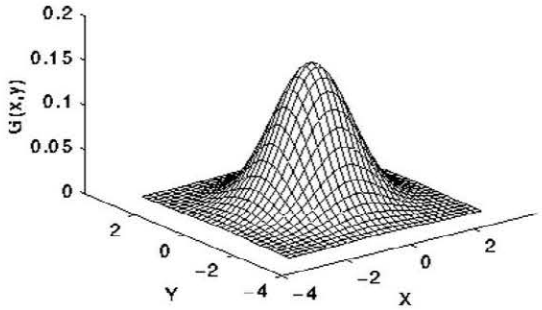
– The Gaussian smoothing operator is a 2-D convolution operator that is used to `blur' images and remove noise (and details).

– Although the objective is removing noise, unwanted side effects are blurring and removal of important information. – The Gaussian function in 2-D has the form:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Preprocessing of Images – image filtering(smoothing)

- 2-D Gaussian function with mean (0,0) and σ =1



- The idea of Gaussian smoothing is to use this 2-D distribution as a `point-spread' function, and this is achieved by convolution

Preprocessing of Images – image filtering(smoothing)

- Discrete approximation to Gaussian function with σ =1.4

$$\text{Gaussian} = \frac{1}{159}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

- convolve(img, oimg, xsize, ysize, gaussian,5,5,115.0);

Preprocessing of Images – image filtering(convolution)

- Implementation of convolution
- – One of the input arrays is a gray level image.
- – The second array is convolution kernel.
- – The convolution is performed by sliding the kernel over the image row by row, starting at the top left corner. – For an image of M*N, and a kernel of J*K the convolution at an image point (s,t) is :

$$c(s,t) = \sum_x \sum_y w(x,y) f(s-x, t-y)$$

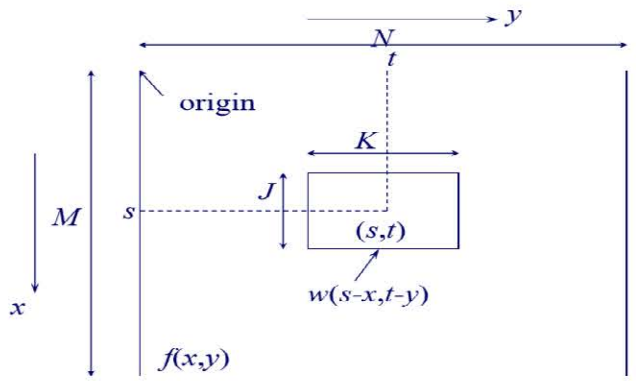
$$c(s,t) = \sum_x \sum_y w(s-x, t-y) f(x,y)$$

where $s = 0,1,2,\ldots,M-1$, $t = 0,1,2,\ldots,N-1$.

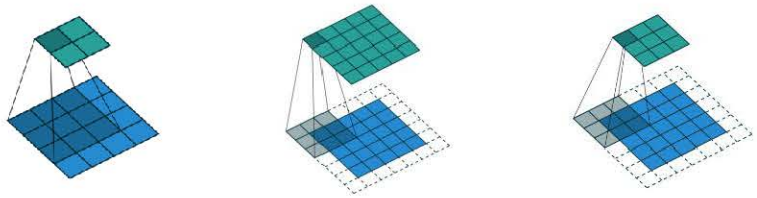
– summation is taken over the image region where w and f overlap.

Preprocessing of Images – image filtering(convolution)

- Illustration of Convolution



Preprocessing of Images – image filtering(convolution)



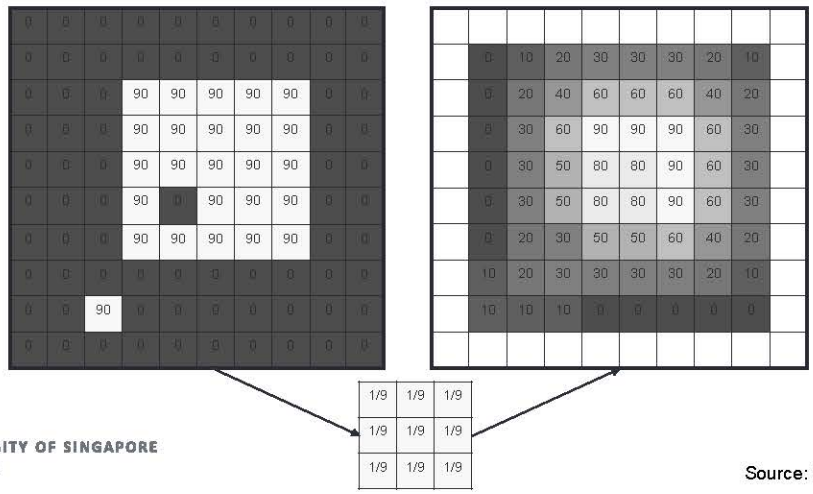
No padding, no strides Half padding, no strides Padding, strides

Convolution on images, strides and padding depends on the required output

Preprocessing of Images – image filtering(convolution)

Moving Average In 2D

$F[x, y]$ $G[x, y]$



Source: S. Seitz

Preprocessing of Images – image filtering(convolution)

- Decompose 2D Gaussian into two separate 1D filters :

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

$$=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}\delta(y)*\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{y^2}{2\sigma^2}}\delta(x)$$

$$G(x,y) = g_x * g_y$$

where $*$ is the convolution operator

$$\therefore\ I(x,y)*G(x,y)=I(x,y)*g_x*g_y=\left[I(x,y)*g_x\right]*g_y$$

Preprocessing of Images – image filtering(convolution)

2-D Gaussian is separable into x and y components.

– Thus 2-D convolution can be performed by first convolving with a 1-D Gaussian in the x direction, and then convolving with another 1-D Gaussian in the y direction.

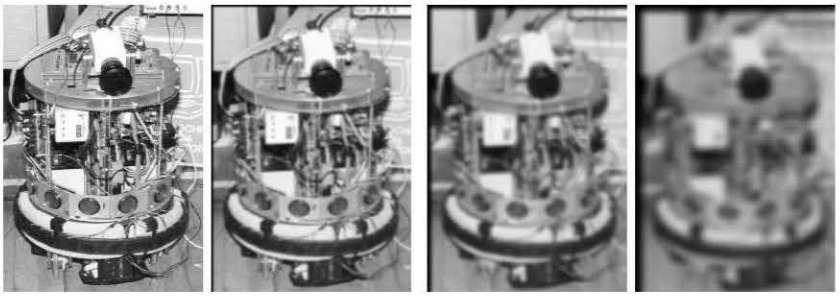
– Therefore,

Gaussian = $\frac{1}{12}$ | 1 | 3 | 4 | 3 | 1 |

convolve(img, tmp,xsize, ysize,gaussian,1,5,10.7); convolve(tmp, oimg,xsize, ysize,gaussian,5,1,10.7);

Preprocessing of Images – image filtering(convolution)

- Examples of Gaussian Smoothing with different σ:



original image | $\sigma = 1.0$ (5×5 kernel) | $\sigma = 2.0$ (9×9 kernel) | $\sigma = 4.0$ (15×15 kernel)

Preprocessing of Images – image filtering(convolution)

Attractive Properties of Gaussian Smoothing

– Rotationally symmetric. Good as we want to smooth the image to the same degree in all directions.

– It has a single lope. This implies that a weighted average is used with decreasing weights for pixels away from the centre pixel.

– The degree of smoothing is controlled by just a single parameter, sigma. Large sigma means a higher degree of smoothing

– Gaussian filter is separable in x and y directions -> significant computational savings.

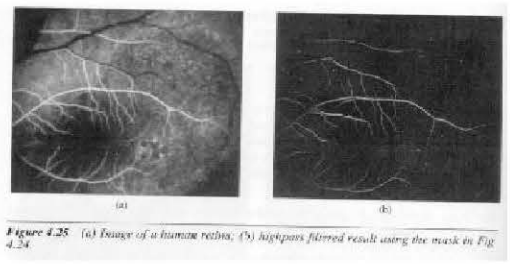
– Freq. Response of the filter has Gaussian shape. Hence, very high freq. Components are eliminated, i.e. Noise cleaning.

Preprocessing of Images – image filtering

High-pass spatial filter

– a high-pass spatial filter can be designed as follows.

-1	-1	-1
-1	8	-1
-1	-1	-1



{Refer to slide (64) Showing Low, high band pass to see the spatial shape of high pass filter}

Image of human retina and its high pass filtered version

Preprocessing of Images – image filtering(noise removal)

Noise Removal

- Image noise can be thermal white noise or impulse noise.
- Lowpass filtering suppresses thermal noise well at the expense of image resolution, i.e. Causing image blur. E.g., Gaussian smoothing which we've already looked at.
- Temporal Averaging reduces image noise without loss of resolution (in the next few slides).
- Median filter reduces impulse noise without loss of resolution (in the next few slides)

Preprocessing of Images – image filtering(noise removal)

- Temporal Averaging
  - Consider that the noise is additive, has zero mean and is uncorrelated,

$$g(x,y) = f(x,y) + \eta(x,y)$$

  - By taking average of M images,

$$\bar{g}(x,y) = \frac{1}{M}\sum_{i=1}^{M} g_i(x,y)$$

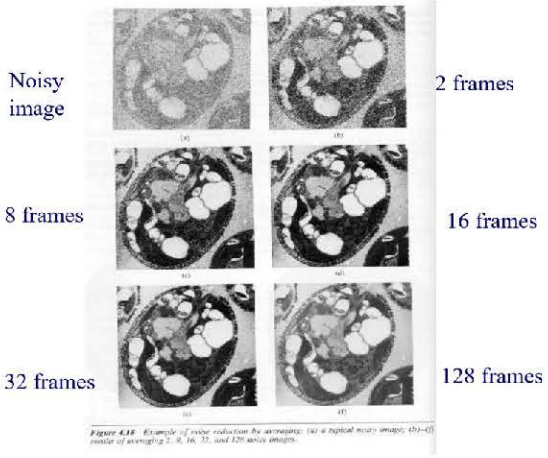
  - then

$$E\{\bar{g}(x,y)\} = f(x,y)$$

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{M}\sigma^2_{\eta(x,y)}$$

  - Hence, noise level is reduced by taking more images.
  - When can we apply temporal averaging?

Preprocessing of Images – image filtering(noise removal)



Noise reduction by averaging

Preprocessing of Images – image filtering(noise removal)

Median filter (Special and Important Case)

– For N being odd, median m = (N+1)/2

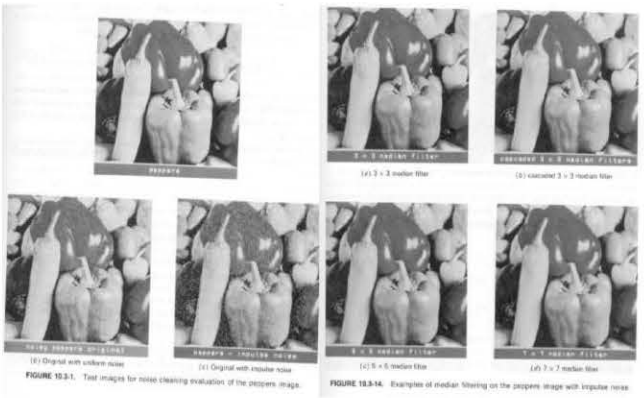
med(R(x))= Rankm(R(x))

– The filter is non-linear

med(R1(x)+R2(x)) != med((R1(x)+med(R2(x))

– median filter does not introduce new intensity values.

– Median filter preserves the location of edges.



Median filtering with impulse noise

03 Pattern extraction

Pattern extraction – template method

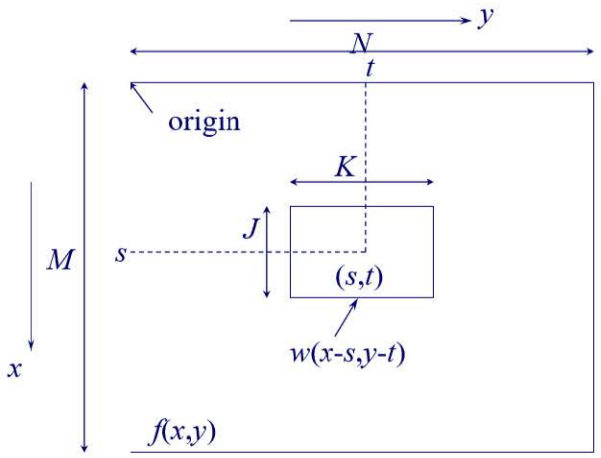
Matching (G & W)

1. Matching by normalized cross correlation • find matches of a sub-image w(x,y) of size J*K within an image f(x,y) of size M*N, by

$$c(s,t) = \frac{\sum_x \sum_y f(x,y)w(x-s,y-t)}{\left\{\sum_x \sum_y [f(x,y)]^2 \sum_x \sum_y [w(x-s,y-t)]^2\right\}^{1/2}}$$

where s =0,1,2,....,M-1, t = 0,1,2,...., N-1.

Pattern extraction – template method



Pattern extraction – template method

Properties:

- the numerator is the cross correlation function.
- the denominator is the normalizer.
- 0 <=c(s,t) <=1
- summation is taken over the image region where w and f overlap.
- The maximum value of c(s,t) appears at the position where w(x,y) best matches f(x,y).
- Advantage : easy to implement.
- Disadvantages : sensitive to image rotation and image scale changes.

Pattern extraction – template method

- Numerical example illustrating normalized cross-correlation:

f =

2	4	66	7	8	9
34	5	6	7	2	4
4	44	5	6	9	10
3	6	7	8	9	7
5	7	3	5	7	5

y

x

w =

5	6	7
44	5	6
6	7	8

y

x

c =

	0.5868	0.2877	0.2712	0.6410	
	0.2879	1.0000	0.5719	0.5872	
	0.2840	0.3309	0.6531	0.6638	

y

x

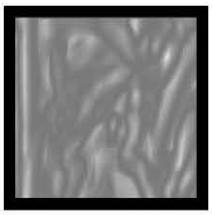
Pattern extraction – template method

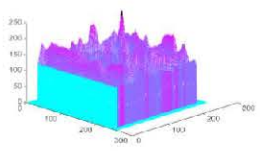
Image

An example of template matching

template

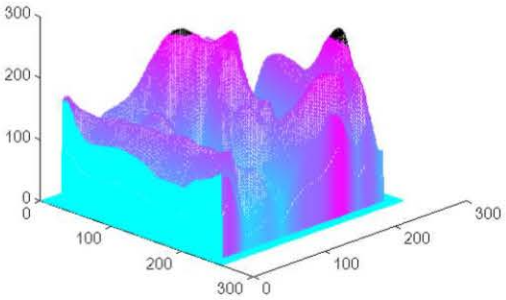
Result of template matching





Pattern extraction – template method

What happens if performing cross correlation without normalization?

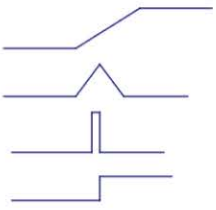


Pattern extraction – edge extraction

- Edge Models
- – Edges are places in the image with strong intensity contrast.
- – Edges often occur at image locations representing object boundaries.

– Types of edges :

- Ramp (1D profile)
- Roof (1D profile)
- Line (1D profile)
- Step (1D profile)

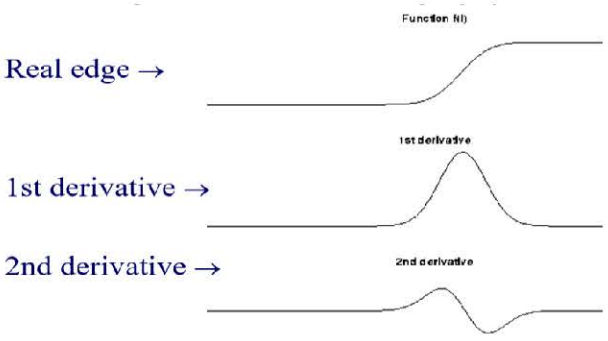


Pattern extraction – edge extraction

The Step Edge Model

– Step edge exist for artificially generated images.

– Real images DO NOT have step edges because anti-aliasing filter used in the imaging system.



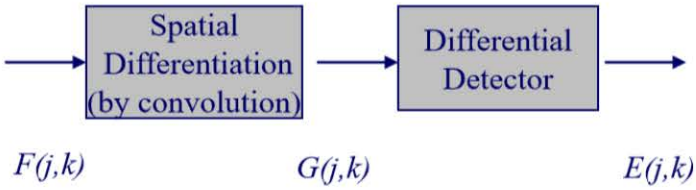
Pattern extraction – edge extraction

The Step Edge Model

– Edge can therefore be located at

- local gradient maximum in the 1st derivative ;
- zero-crossing in the 2nd derivative.

– Edge can therefore be detected using differential operators.

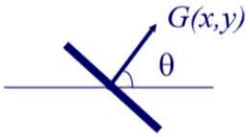


Pattern extraction – edge extraction

- First derivative operator
- – Also known as Gradient operator.
- – For an edge in a 2D image,

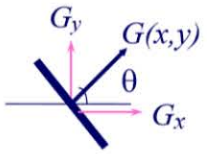
$$G(x,y) = \frac{\partial F(x,y)}{\partial x}\cos(\theta) + \frac{\partial F(x,y)}{\partial y}\sin(\theta)$$

- where G(x,y) is the gradient normal to the edge.



Pattern extraction – edge extraction

- First derivative operator
- – Usually, we compute



$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

where G_x is the horizontal gradient,

G_y is the vertical gradient.

$$|\nabla f| = \left[G_x^2 + G_y^2\right]^{1/2}$$

$$|\nabla f| \approx |G_x| + |G_y| \quad ; \theta(x,y) = \tan^{-1}\left(G_y / G_x\right)$$

Pattern extraction – edge extraction

- Robert's cross gradient operator

+1	0
0	-1

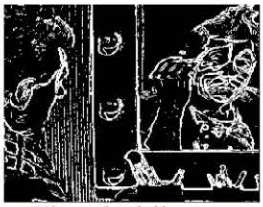
Gx

0	+1
-1	0

Gy

original

Convolved with Robert's operator



Thresholding on Robert's output

Pattern extraction – edge extraction

- Sobel's gradient operator

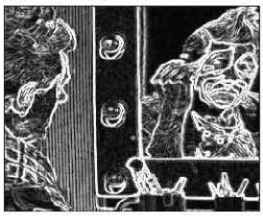
-1	0	+1
-2	0	+2
-1	0	+1

Gx

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

Gy

original



Convolved with Sobel's operator

Pattern extraction – edge extraction

Gaussian smoothing + gradient operator

Gaussian smoothing is performed to suppress image noise before the image is differentiated.

$$G(x,y) = \nabla[g(x,y) * I(x,y)] \quad \leftarrow \text{Convolution first}$$
$$= \nabla g(x,y) * I(x,y) \quad \leftarrow \text{Differentiation first}$$

– Therefore, edge can be detected by convolving the image with the first derivative of Gaussian. – The 2D first derivative of Gaussian is also separable into 1D filters.

– Combining Gaussian smoothing with differentiation reduces computation.

Pattern extraction – edge extraction

Second derivative operator

– Laplacian operator:

- The Laplacian L(x,y) of an image at I(x,y) is:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

– Laplacian kernels: (basic kernel [1 -2 1])

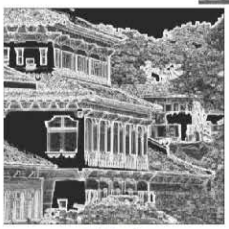
0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

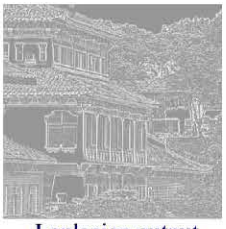
– Laplacian operator usually produces closed contour

Pattern extraction – edge extraction

original



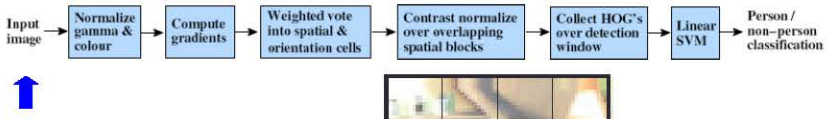
Sobel output

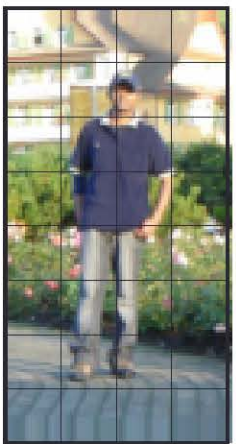


Laplacian output

Pattern extraction – HoG

- Histogram of gradient (HoG)

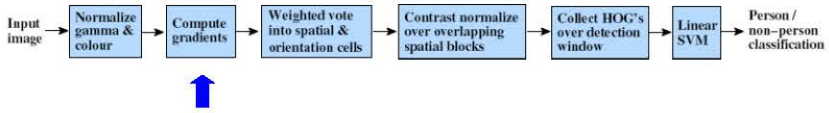




Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Pattern extraction – HoG

- Histogram of gradient (HoG)

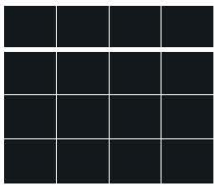


Notes on the orientation:
1) θ will be the same when(Iy=1,Ix=1) and (Iy=-1,Ix=-1). Hence , dealing with this, we define:
$\theta = \tan^{-1}\left(\frac{|Iy|}{|Ix|}\right)$; and keep the signs of Ix & Iy

2) Hence, 0<= θ <=360

Pattern extraction – HoG

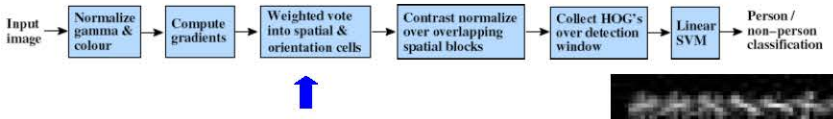
- How can we obtain the histogram of the oriented gradients?
- Two important concepts:
- Block operation & histogram
- Block Operation: Divide the input image into blocks:
- Each block can be 4*4 or 16*16 pixels (depends on the image size & application)



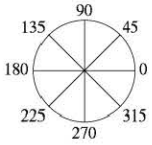
Pattern extraction – HoG

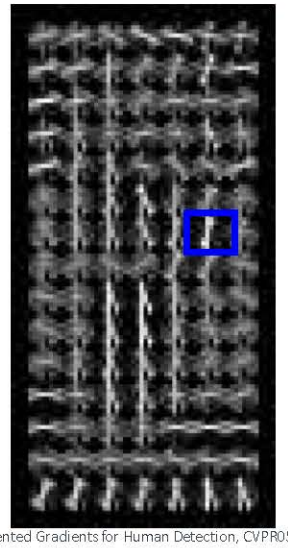
- For each pixel in the block we will have one set of oriented gradient information (θ,M).
- If we have 4*4 pixels , then we have 16 sets of in (θ,M), where some of them might have the same/similar values.

Pattern extraction – HoG



- Histogram of gradient orientations
  -Orientation

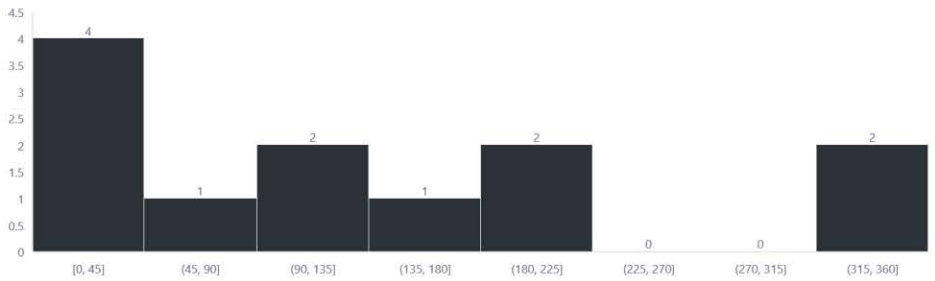




Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Pattern extraction – HoG

- Histogram of Orientated Gradients
- 1. For each block (cell), we can group the gradients (16 of them) using a histogram with 8 bins (8 orientations)



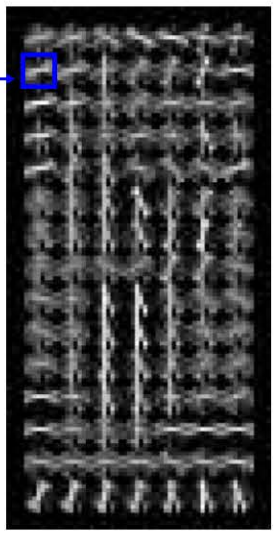
Pattern extraction – HoG

- The magnitude influences the height of each bin.
- A HoG descriptor:
- Concatenates the histograms of the all blocks (or cells) in one vector.

Pattern extraction – HoG

8 orientations

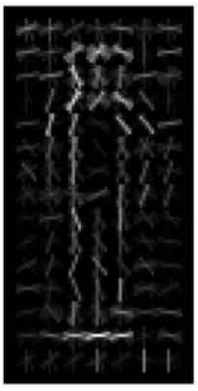
X=

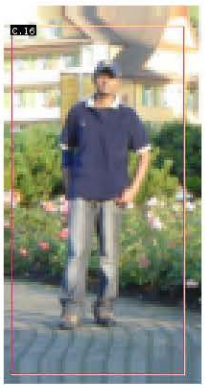


$\in R^{840}$

15x7 cells

Pattern extraction – HoG





$0.16 = w^T x - b$

$sign(0.16) = 1$

$=>$ pedestrian

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

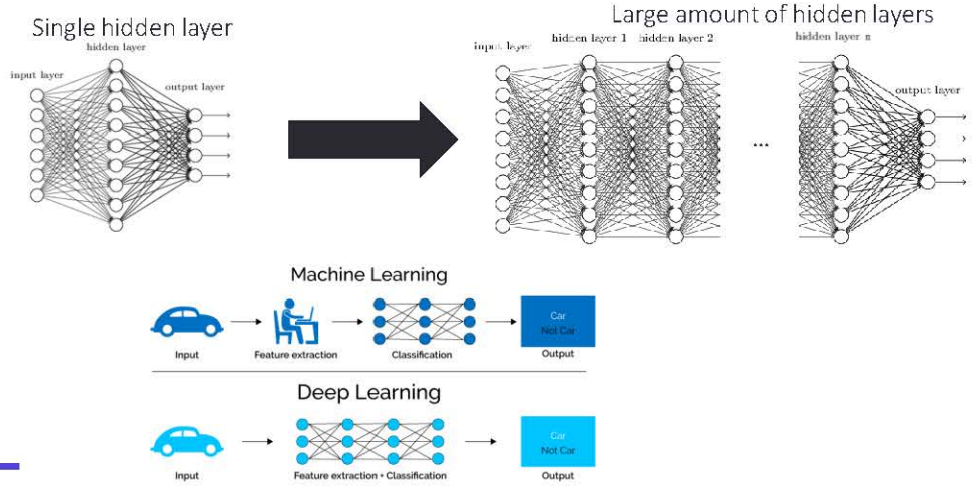
04 Convolution Neural Network

HOG + SVM: **complicated and not straightforward**

Can we combine the Feature Extraction and Classification to be ONE algorithm?

Most of human brain intelligence may be due to **one learning algorithm**.
Neural Network could do both **Mapping and Classifying.**

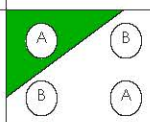
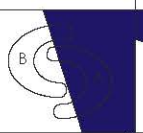
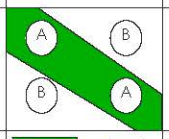
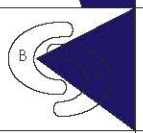
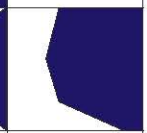
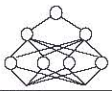
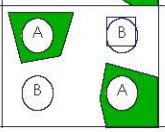
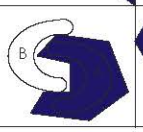
NN Structure



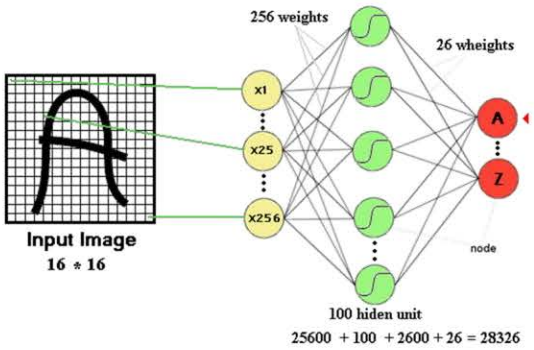
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

Image feature extraction

Behavior of multilayer neural networks

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyper plane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

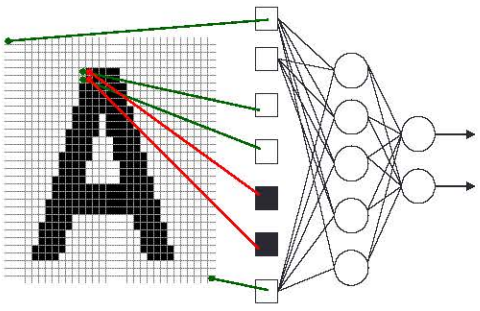
One or more hidden layers



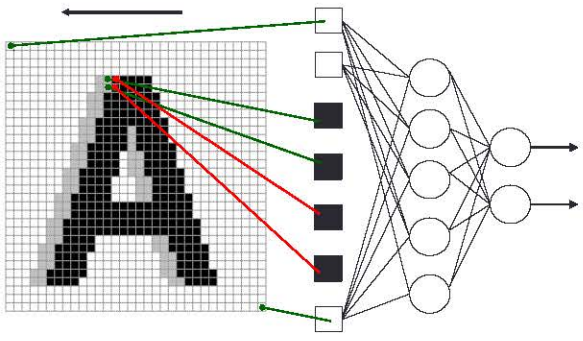
Fully connected network of sufficient size can produce outputs that are invariant with respect to such variations.

Training time
Network size
Free parameters

Little or no invariance to shifting, scaling, and other forms of distortion



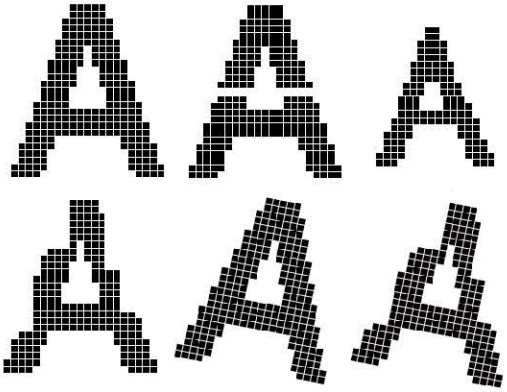
Shift left



154 input change from 2 shift left
77 : black to white
77 : white to black

Image feature extraction

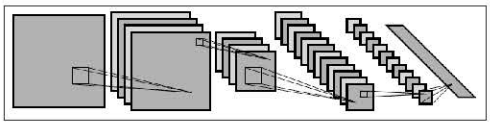
scaling, and other forms of distortion

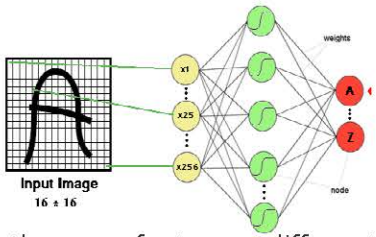


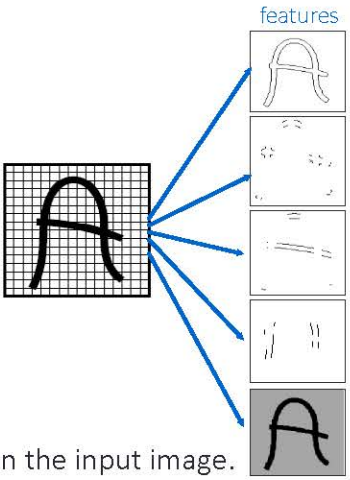
the topology of the input data is completely ignored
work with raw data.

Image feature extraction

What we can do with extracting features?
Use image convolution as input of NN

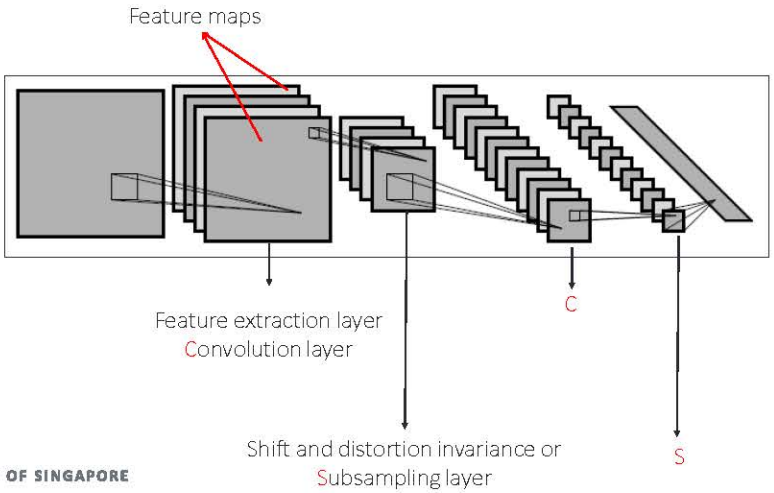




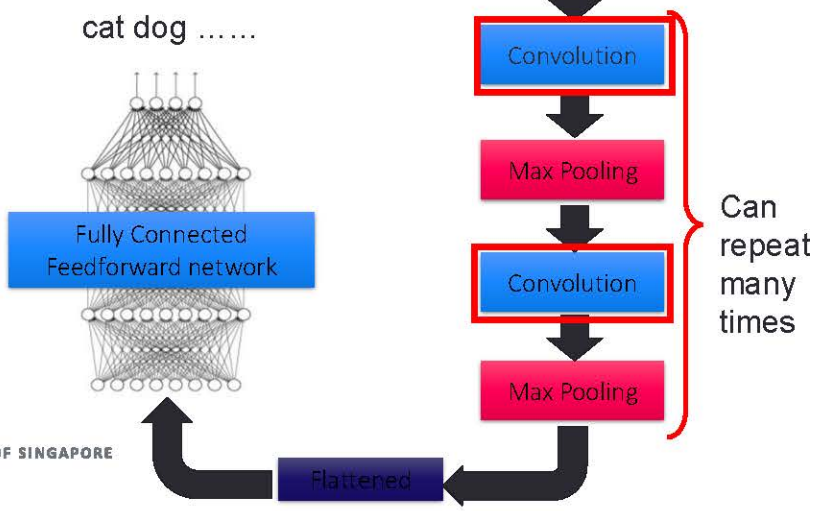


Detect the same feature at different positions in the input image.

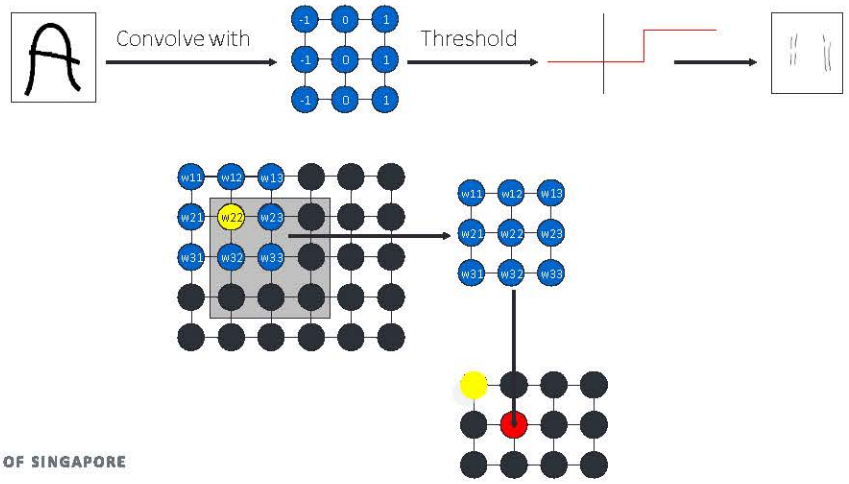
CNN introduced by Yann LeCun



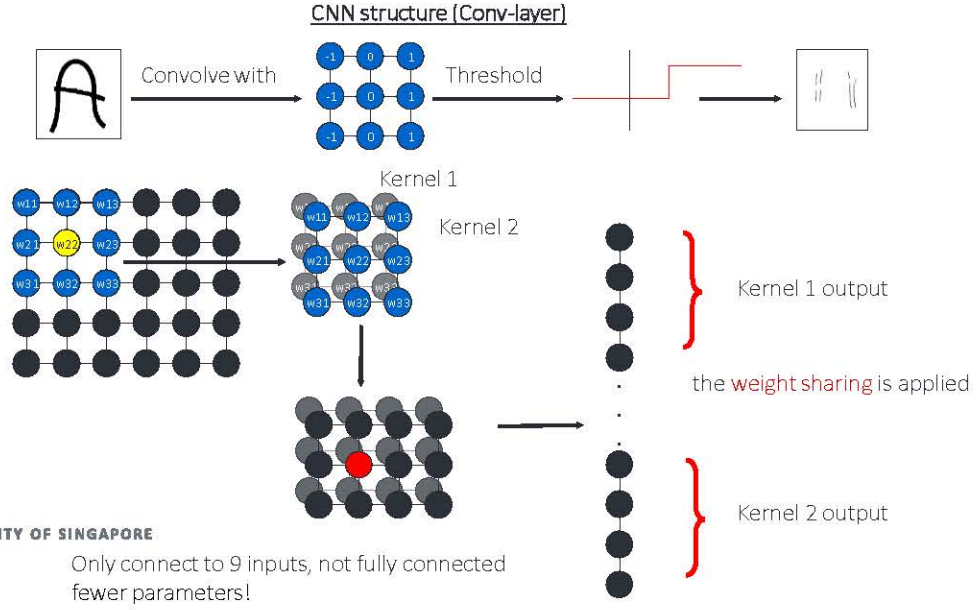
CNN structure



CNN structure (Conv-layer)



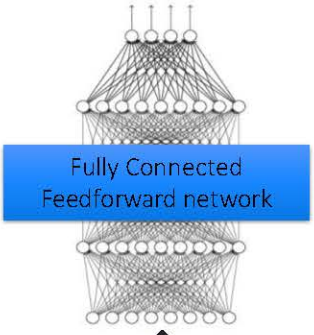
CNN structure (Conv-layer)



Only connect to 9 inputs, not fully connected
fewer parameters!

CNN structure

cat dog ……



Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flattened

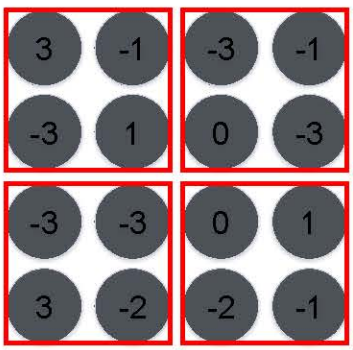
CNN structure (Max pooling)

Filter 1

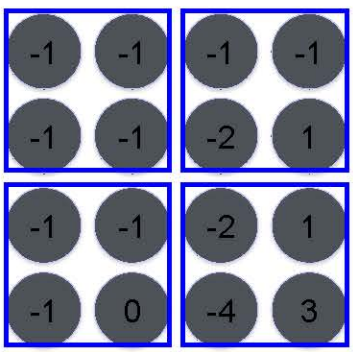
1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1



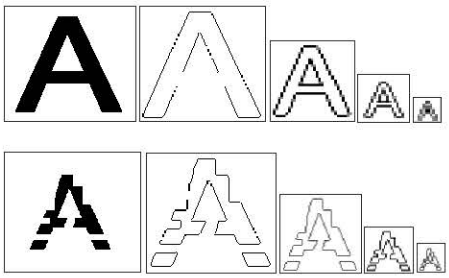
Only reserve local maximal value.



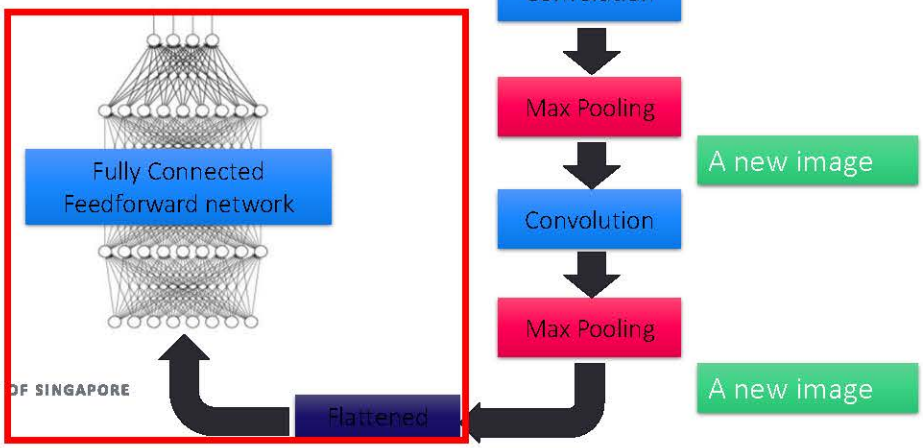
CNN structure (Max pooling)

The Max pooling (subsampling) layers reduce the spatial resolution of each feature map

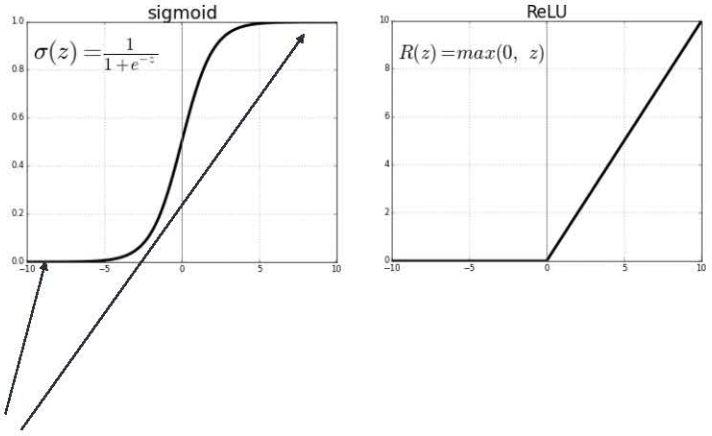
By reducing the spatial resolution of the feature map, a certain degree of shift and distortion invariance is achieved.



CNN structure

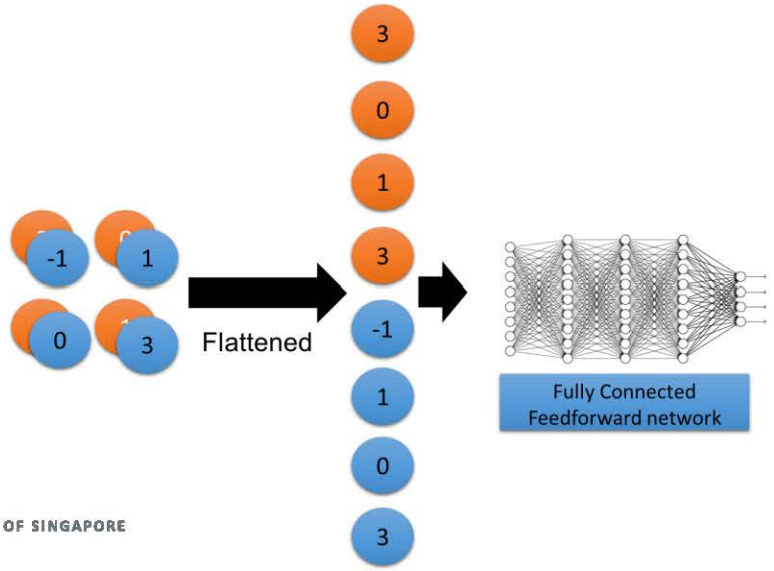


CNN structure (activation function)



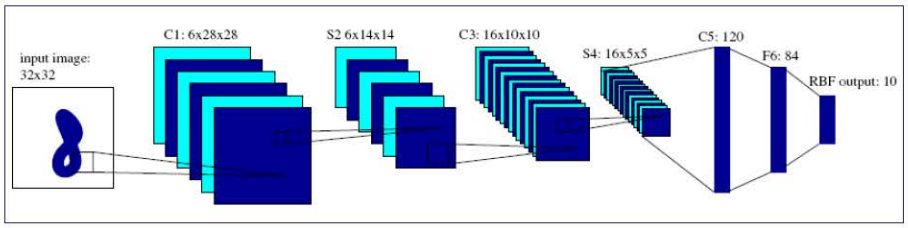
When magnitude of input very large or very small, gradient to be 0!
In deep learning, this happens a lot.

CNN structure (output)

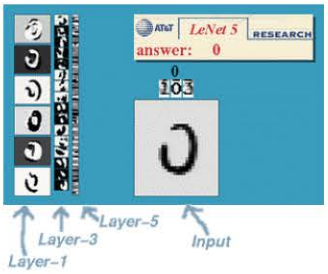


CNN case

LeNet5

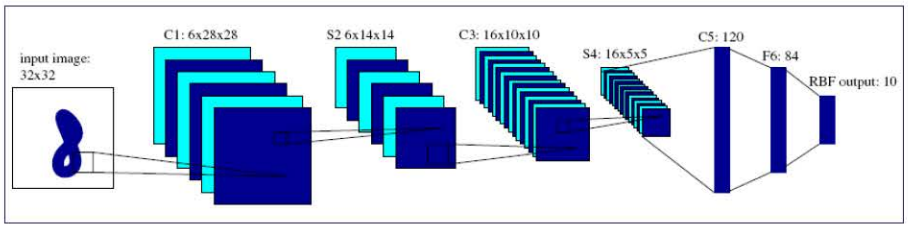


- About 187,000 connection.
- About 14,000 trainable weight.



CNN case

LeNet5



Structure: http://scs.ryerson.ca/~aharley/vis/conv/

Training: https://cs.stanford.edu/people/karpathy/convnetjs/index.html

05 Introduction to Deep Learning

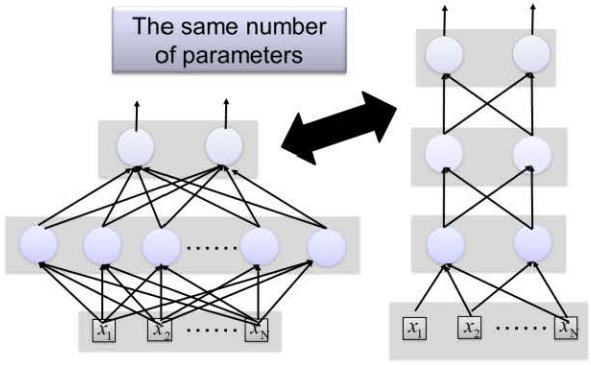
Deep learning means larger number of hidden layers,
It shows decent performance especially in **CNN**!

BUT Why **deep is better**?

Deeper is better

1. Experiments show deeper is better

Fat + Short v.s. Thin + Tall



Which one is better?

Deeper is better

1. Experiments show deeper is better

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

It is like conditional probability.

1-3-3-1 can represent 3 x 3 situation, but only need 6 parameters.

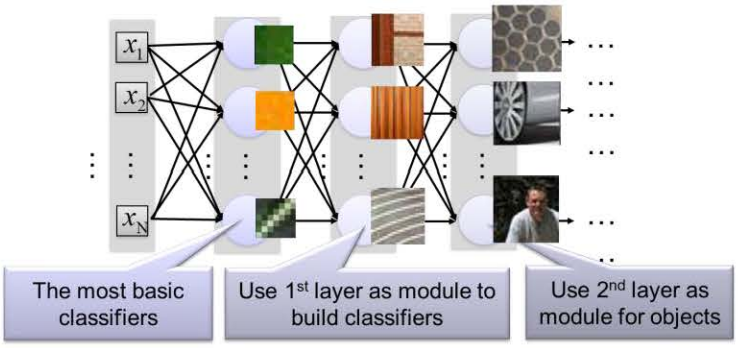
1-9-1 have 9 parameters

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Deeper is better

1. Experiments show deeper is better

- Levels of features



Reference: Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014* (pp. 818-833)

It is like a cascade classification process, every step narrow down the searching range

Deeper is better

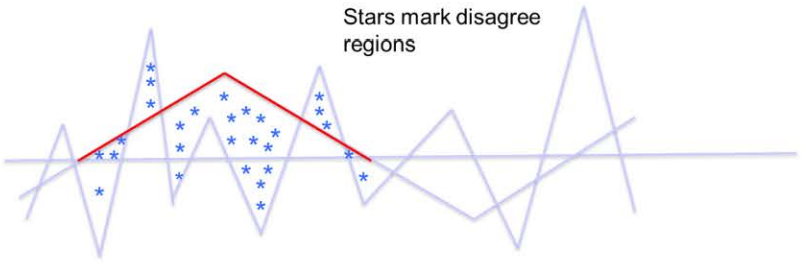
2. Theoretical proof: Deeper is better

M. Telgarsky: The benefit of depth in neural networks, 2016.

We give an informal argument of the Telgarsky's proof.

Claim 1. Few oscillations can't fit many oscillations.

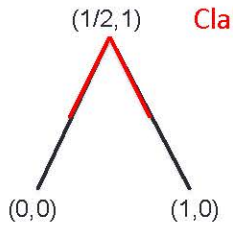
Proof by picture



Deeper is better

2. Theoretical proof: Deeper is better

M. Telgarsky: The benefit of depth in neural networks, 2016.



Claim 2. ReLU can make exponentially many oscillations

ReLU(x) := max {0,x}, and
Let h(x) := ReLU(ReLU(2x) – ReLU(4x-2))

h(x)

$$h(x) = \begin{cases} 2x & x \in [0, \tfrac{1}{2}] \\ 2(1-x) & x \in [\tfrac{1}{2}, 1] \\ 0 & \text{otherwise} \end{cases}$$

h has 1 peak → h^k has 2^{k-1} peaks.

Deeper is better

2. Theoretical proof: Deeper is better

M. Telgarsky: The benefit of depth in neural networks, 2016.

Claim 3. multiple layers implies multiple oscillations.

There exists a function that can be learned by a "deep" neural network with a polynomial number of nodes, but it needs exponentially many nodes for any "shallow" neural network.

Deeper is better (experiment)

1. ReLu: Why we use it? Look at the classification edge.
2. Hidden layers: muti-layer perceptron topology study

https://playground.tensorflow.org/

Deep learning structure

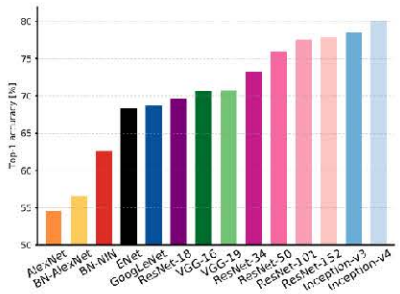


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.

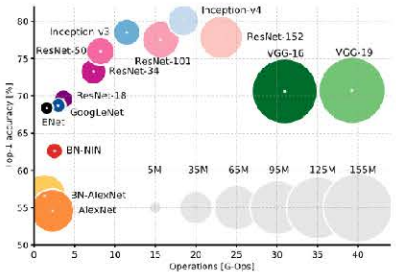


Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs.

Case Study: VGGNet

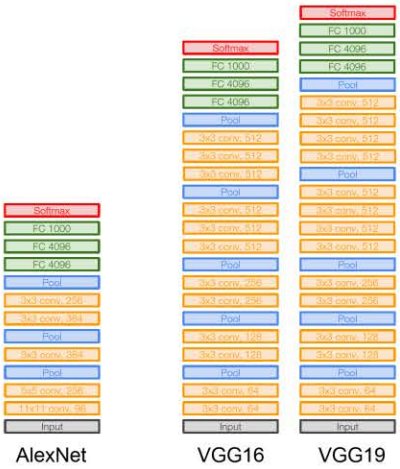
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



Case Study: VGGNet

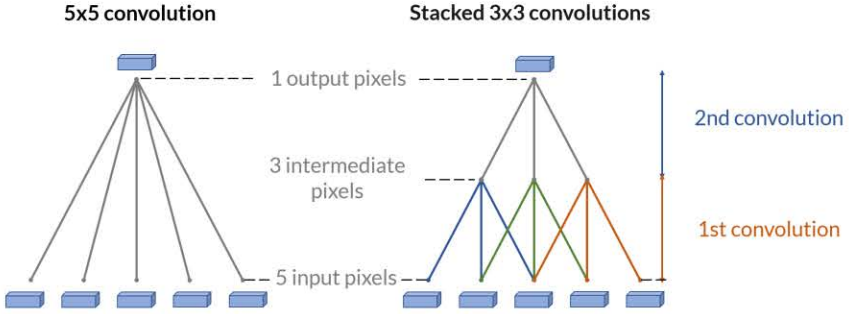
[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3x3 conv (stride 1) layers
has same **effective receptive field** as
one 7x7 conv layer

Q: What is the effective receptive field of
three 3x3 conv (stride 1) layers?

Deep learning (VGG case)



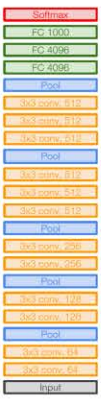
Why smaller filter?

Think about a pixel output, 5x5 kernel means 5x5 pixels' information is included in this 1 pixel.
However, two 3x3 stack kernels also achieve this,
But fewer parameters 3x3+3x3 = 18 < 5x5

Deep learning (VGG case)

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



VGG16

Deep learning (VGG case)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Unfortunately, there are two major drawbacks with VGGNet:

1.It is *painfully slow* to train.
2.The network architecture weights themselves are quite large (in terms of disk/bandwidth).

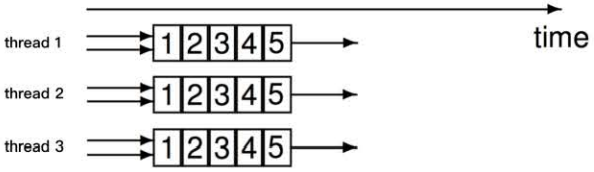
Due to its depth and number of fully-connected nodes, VGG is over 533MB for VGG16 and 574MB for VGG19. This makes deploying VGG a tiresome task.
We still use VGG in many deep learning image classification problems; however, smaller network architectures are often more desirable (such as SqueezeNet, GoogLeNet, etc.).

Deep learning (Hardware)

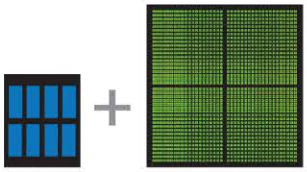
Single thread and Multithreading

For single-core CPUs, originally, each thread completed one operation before the next start to avoid complexity of pipeline overlaps

For multithreading, NVIDIA have now relaxed this, so each thread can have multiple independent instructions overlapping. And multiple cores can parallel different threads.



Deep Learning: Graphic Process Units



CPU vs GPU

	Cores	Clock Speed	Memory	Price	Speed
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	$339	~540 GFLOPs FP32
GPU (NVIDIA GTX 1080 Ti)	3584	1.6 GHz	11 GB GDDR5 X	$699	~11.4 TFLOPs FP32
TPU NVIDIA TITAN V	5120 CUDA, 640 Tensor	1.5 GHz	12GB HBM2	$2999	~14 TFLOPs FP32 ~112 TFLOP FP16
TPU Google Cloud TPU	?	?	64 GB HBM	$6.50 per hour	~180 TFLOP

CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks

GPU: More cores, but each core is much slower and "dumber"; great for parallel tasks

TPU: Specialized hardware for deep learning

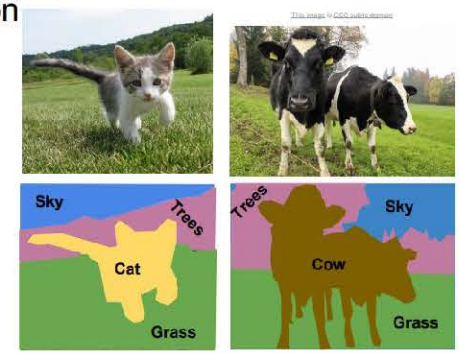
06 Detection & Mask RCNN

Deep Learning(RCNN and Mask-RCNN: Task)

Semantic Segmentation

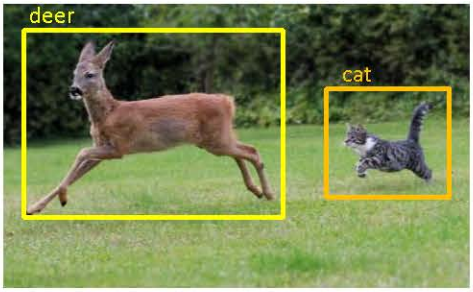
Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels

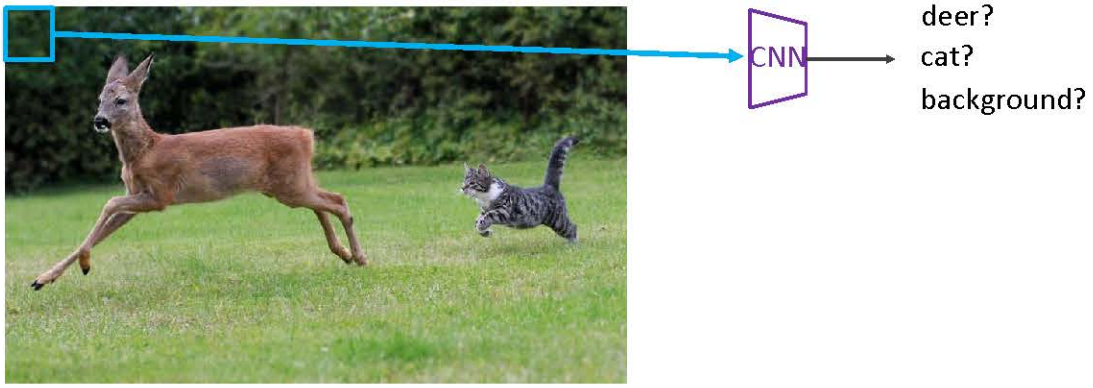


Deep Learning(RCNN and Mask-RCNN)

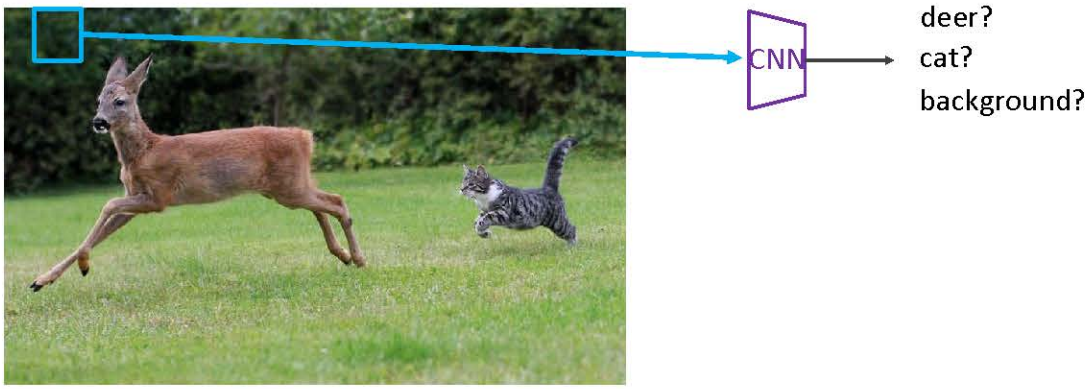
Traditional way to detect object: Sliding Windows



Traditional way to detect object: Sliding Windows



Traditional way to detect object: Sliding Windows

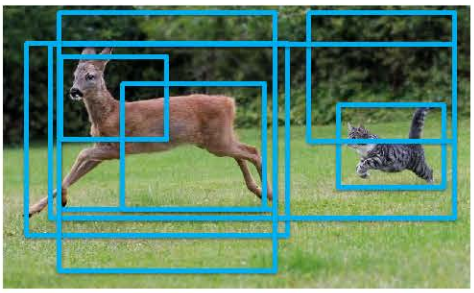


Traditional way to detect object: Sliding Windows

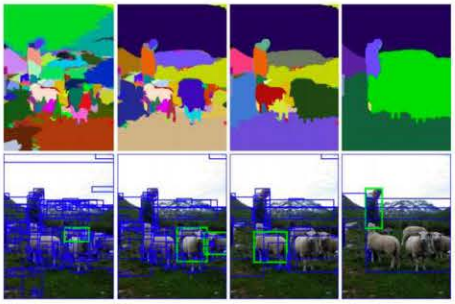
Traditional way to detect object: Sliding Windows

TOO MUCH candidates for running high-cost CNN!

Deep Learning(RCNN and Mask-RCNN)



Use simple method find box proposals

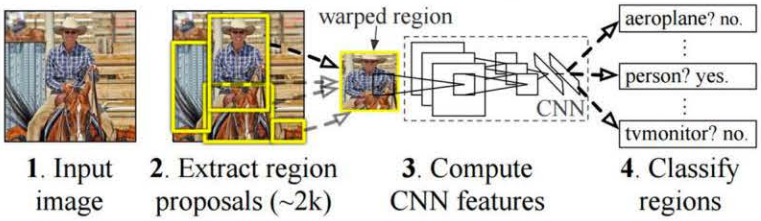


Selective Search find proposal box

van de Sande et al. ICCV 2011

Deep Learning(RCNN and Mask-RCNN)

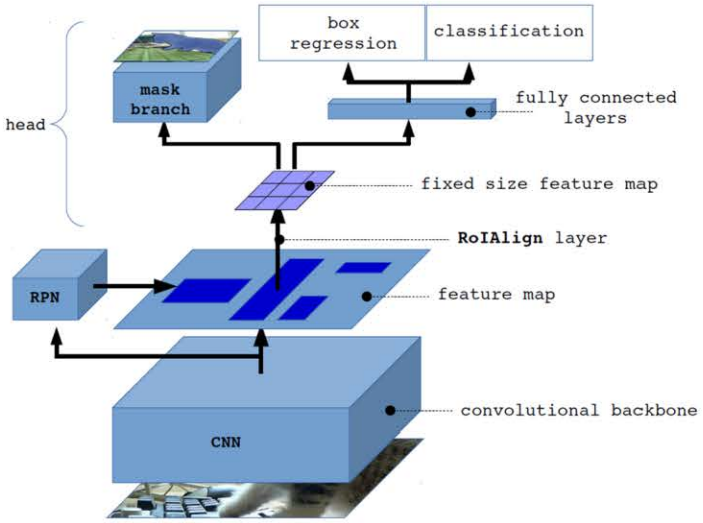
R-CNN: *Regions with CNN features*



Rich feature hierarchies for accurate object detection and semantic segmentation.

Girshick et al. CVPR 2014.

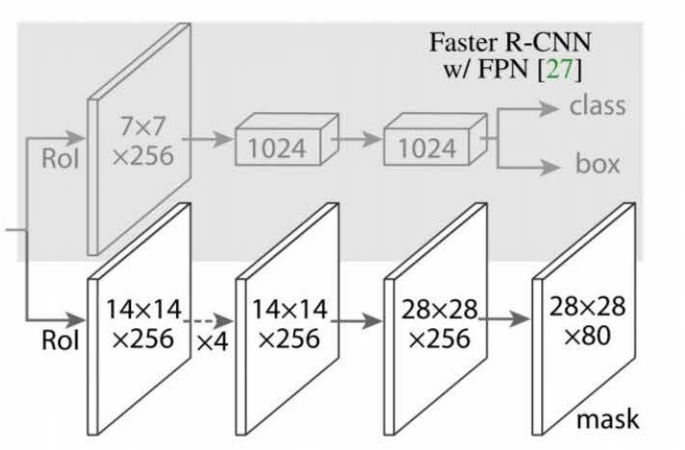
Mask R-CNN: overview



Network Architecture

- Can be divided into two-parts:
  - Backbone architecture : Used for feature extraction
  - Network Head: comprises of object detection and segmentation parts
- Backbone architecture:
  - ResNet
  - ResNeXt: Depth 50 and 101 layers
  - Feature Pyramid Network (FPN)
- Network Head: Use almost the same architecture as Faster R-CNN but add convolution mask prediction branch

FCN Mask Head



Loss Function

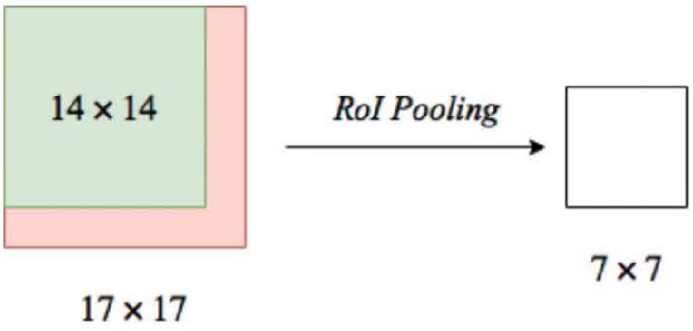
$$L = L_{cls} + L_{box} + L_{mask}$$

- Loss for classification and box regression is same as Faster R-CNN
- To each map a per-pixel sigmoid is applied
- The map loss is then defined as average binary cross entropy loss
- Mask loss is only defined for the ground truth class
- Decouples class prediction and mask generation
- Empirically better results and model becomes easier to train

$$J = -\frac{1}{N}(\sum_{i=1}^{N} \mathbf{y_i} \cdot log(\hat{\mathbf{y}}_\mathbf{i}))$$

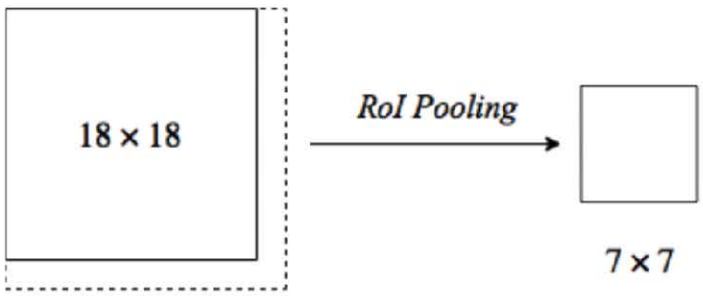
RoI Pooling: Stride is quantized.

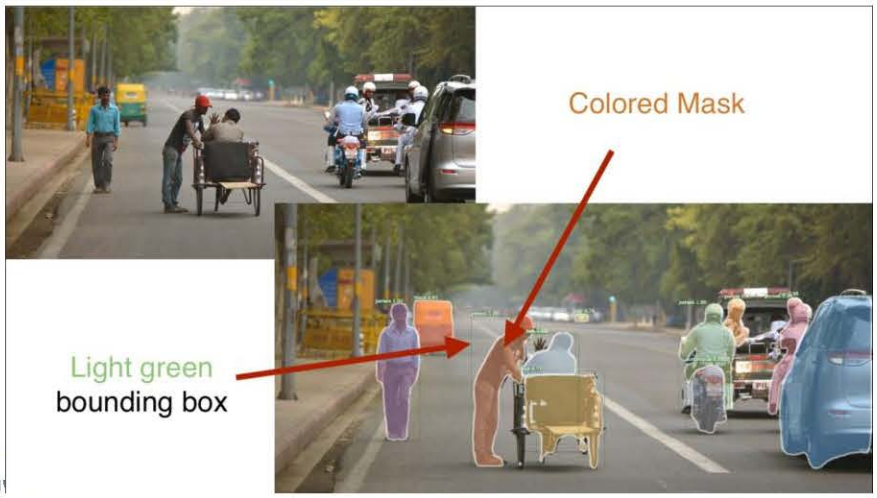
$$stride = \frac{17}{7} = 2.42 \qquad stride_{RoIPool} = \lfloor 2.42 \rfloor = 2$$



RoI Pooling: Stride is quantized.

$$stride = \frac{18}{7} = 2.57 \qquad stride_{RoIPool} = [2.57] = 3$$





Mask R-CNN for Human Pose Estimation

- Model keypoint location as a one-hot binary mask
- Generate a mask for each keypoint types
- For each keypoint, during training, the target is a $m \times m$ binary map where only a single pixel is labelled as foreground
- For each visible ground-truth keypoint, we minimize the cross-entropy loss over a m^2-way softmax output