BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS (CNN) 1-D

Dr. Kamlesh Tiwari

Motivation to Convolution

Suppose you have a noisy sensor to have a measurement

How could you estimate the actual reading

1. Average of some readings
2. in local neighbourhood
3. What about weighted average?

$$s_t = \sum_{i=0}^{\infty} x_{t-i} w_{-i}$$

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0					
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5					
X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80					

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

Here things are in one dimension only.

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS (CNN) 2-D

Dr. Kamlesh Tiwari

Motivation to Convolution

Suppose you have a noisy sensor to have a measurement

How could you estimate the actual reading

1. Average of some readings
2. in local neighbourhood
3. What about weighted average?

$$s_t = \sum_{i=0}^{\infty} x_{t-i} w_{-i}$$

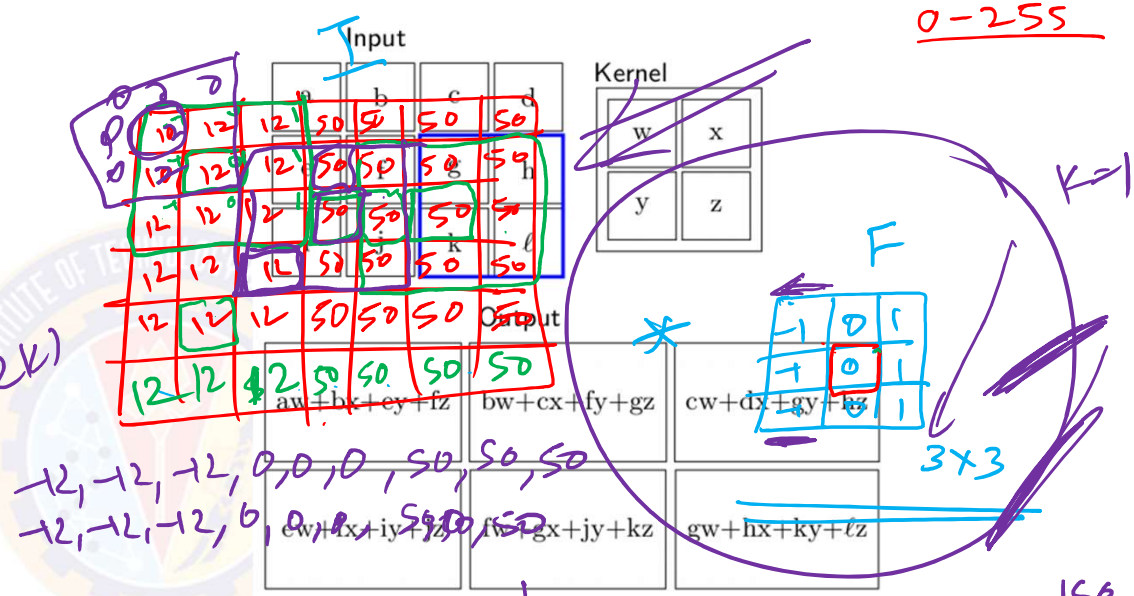
	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0					
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5					
X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
S							1.80					

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

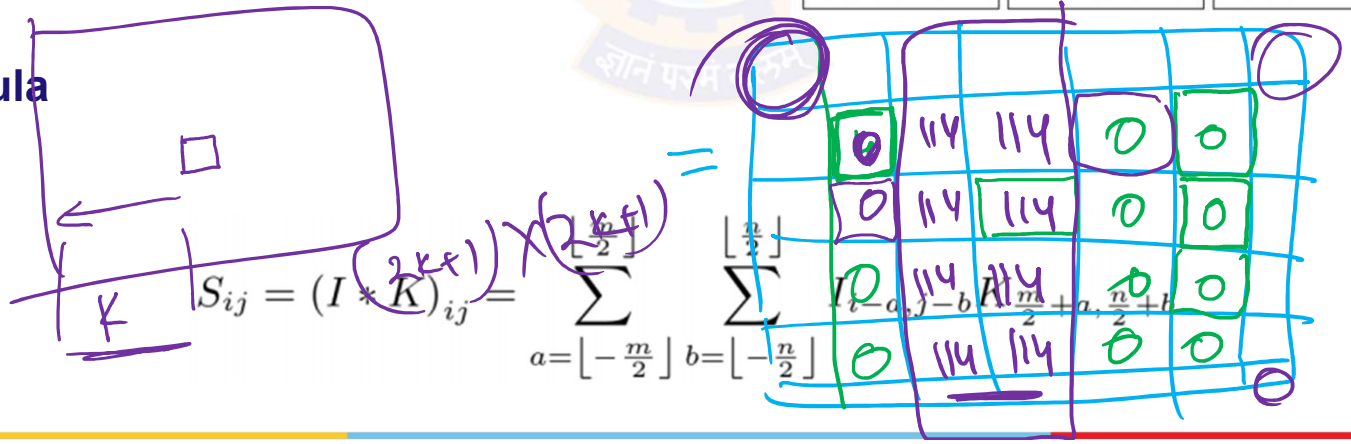
Here things are in one dimension only.

Image is a 2D signal

$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a,j+b} K_{a,b}$$



General formula



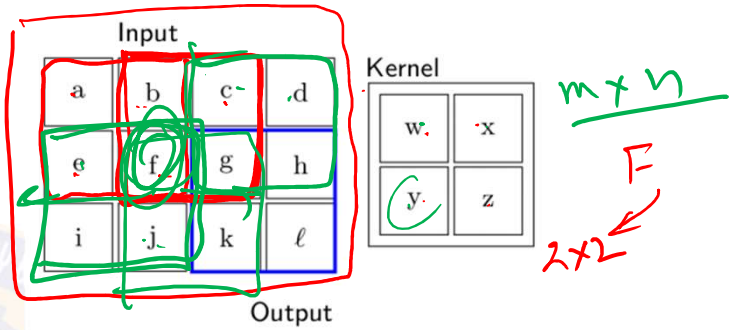
$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a,\frac{n}{2}+b}$$

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS (CNN) 2-D

Dr. Kamlesh Tiwari

Image is a 2D signal



$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a,j+b} K_{a,b}$$

General formula

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a,\frac{n}{2}+b}$$

Image is a 2D signal

$$* \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

blurs the image

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

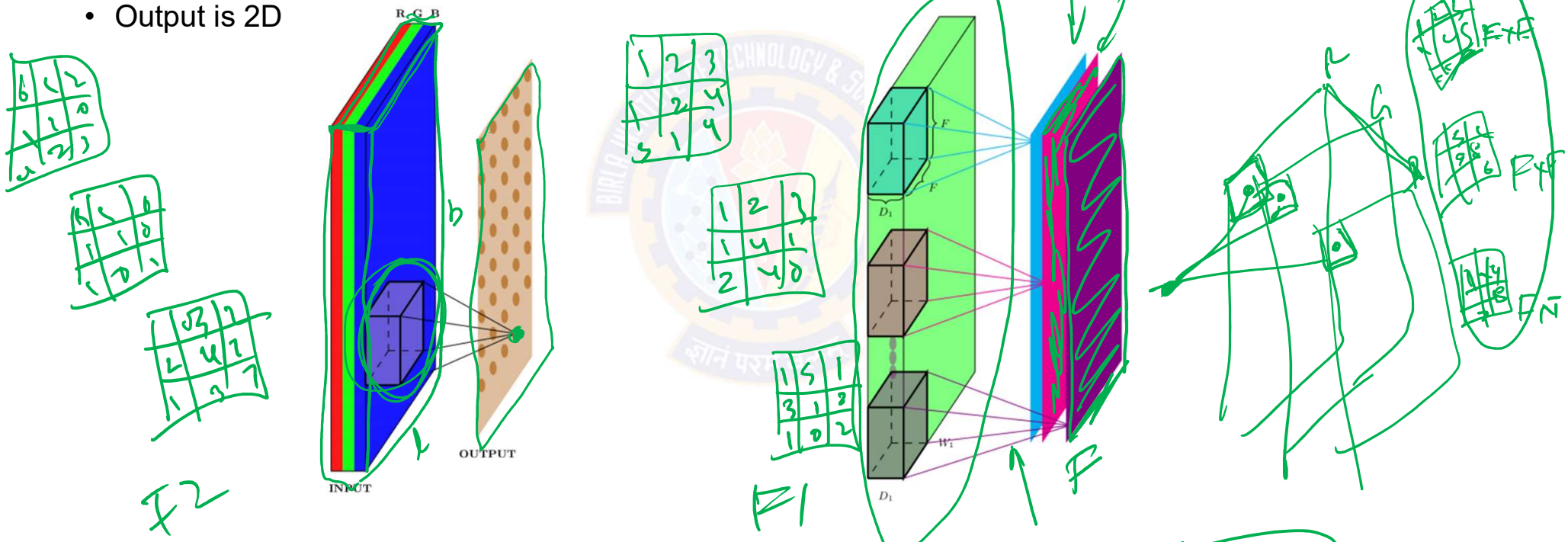
sharpens the image

$$* \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

detects the edges

Filter in 3D

- It would refer to volume. Assume the filter extends to the depth
- Output is 2D



- Apply multiple filters to get multiple feature maps

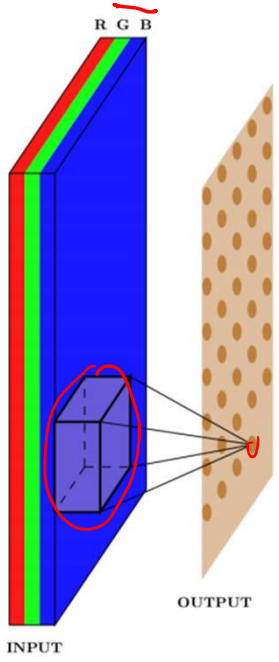
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

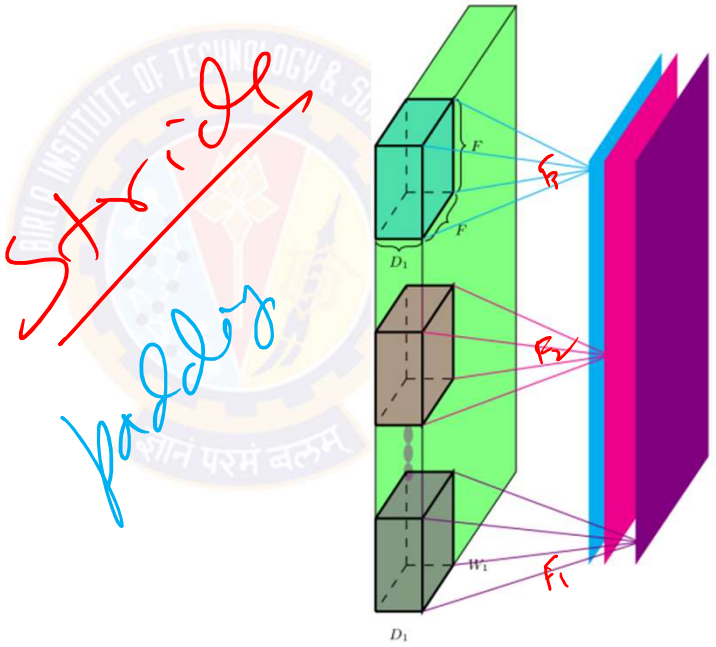
CONVOLUTIONAL NEURAL NETWORKS (CNN) 2-D

Dr. Kamlesh Tiwari

Filter in 3D

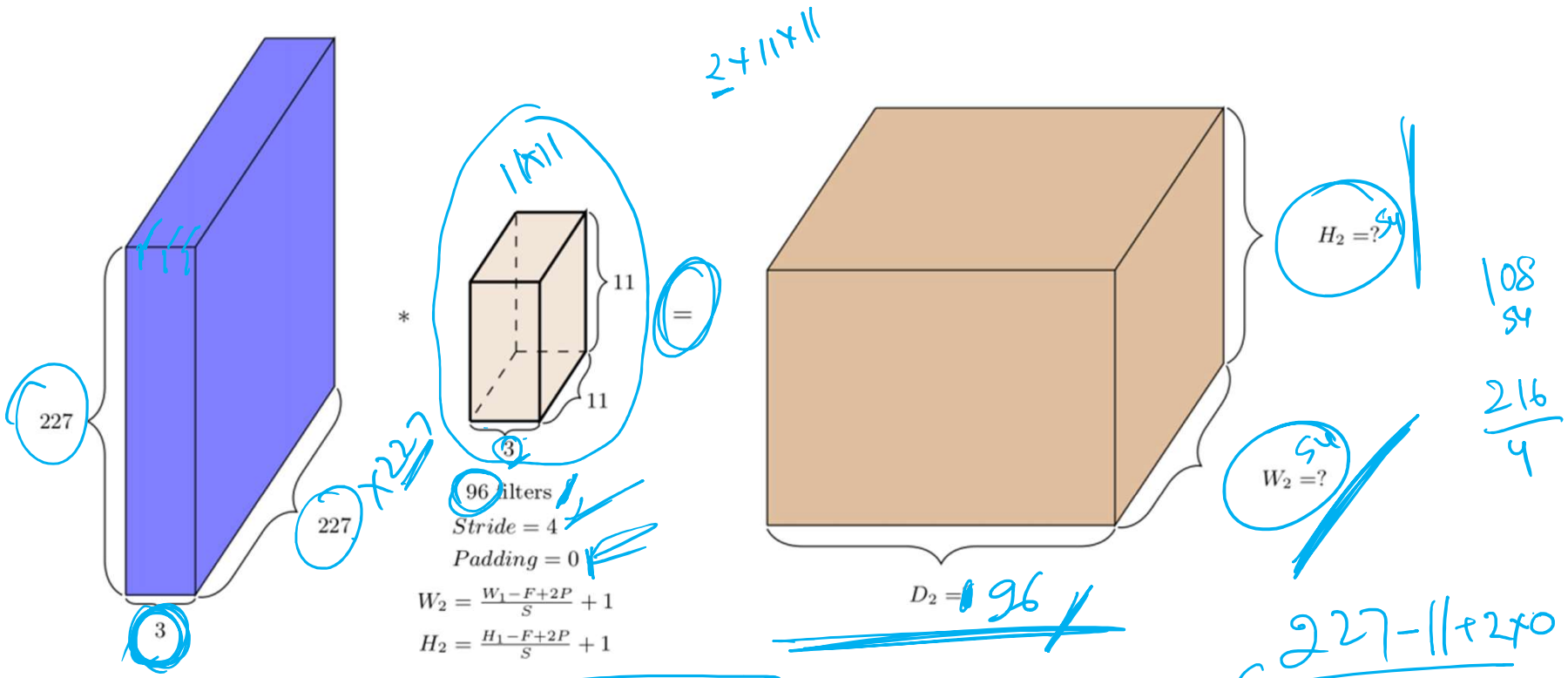
- It would refer to volume. Assume the filter extends to the depth
- Output is 2D



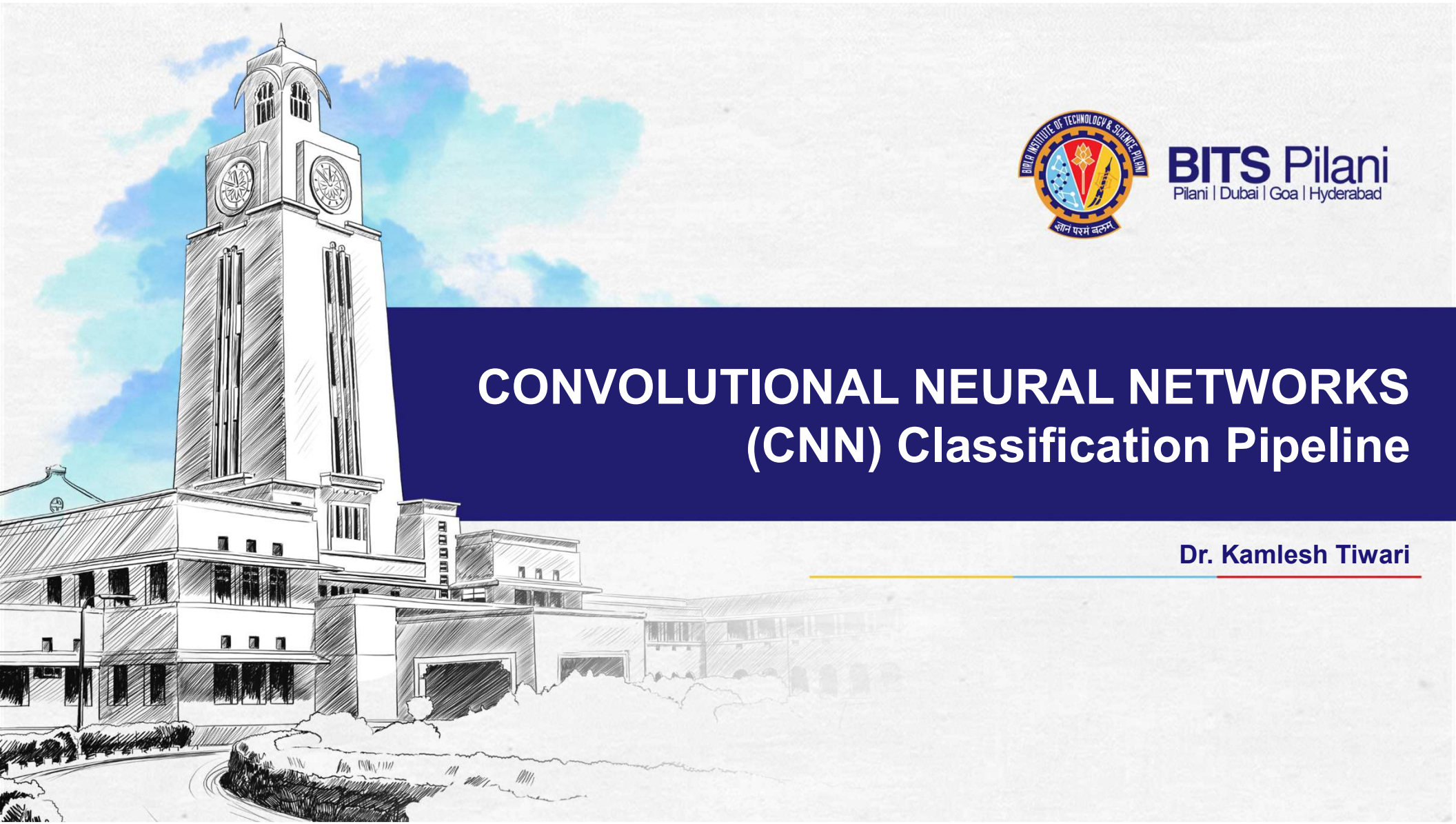


- Apply multiple filters to get multiple feature maps

Filters, Padding and Stride



$$H, W = \frac{H - F + 2P}{S} + 1, \frac{S - F + 2P}{S} + 1$$

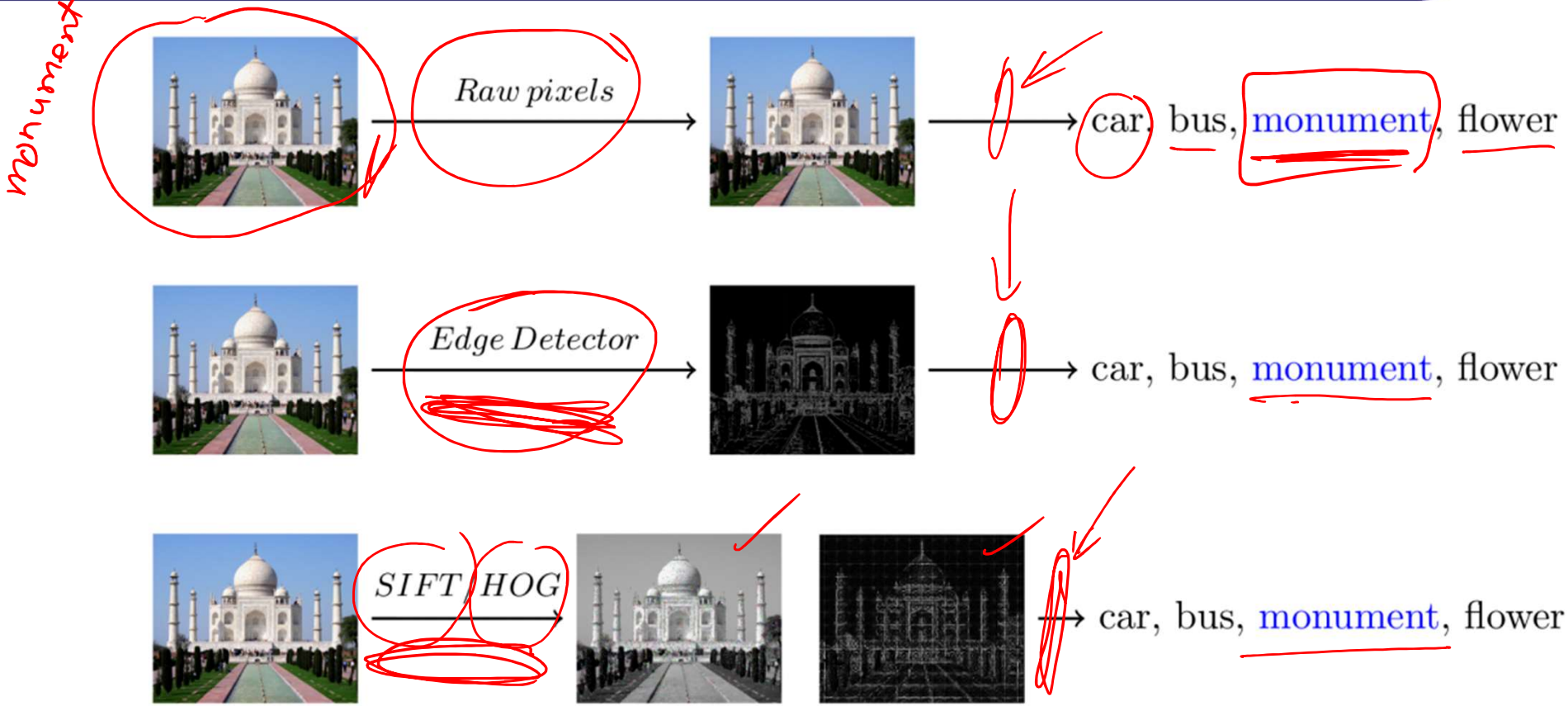


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS (CNN) Classification Pipeline

Dr. Kamlesh Tiwari

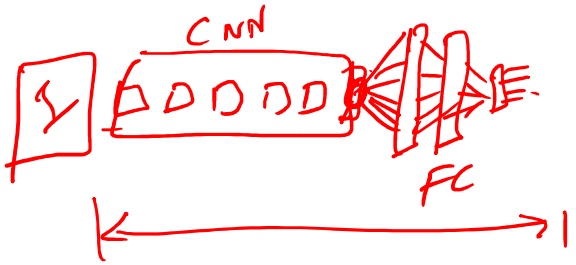
Classification Pipeline



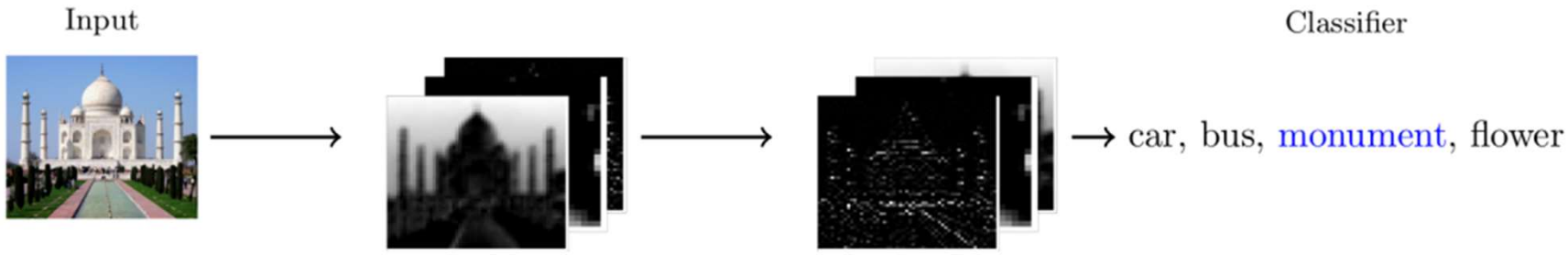
- Where is learning? (hand craft features, then learn weights for classification). One can see feature as a convolution.

Automate feature kernel discovery

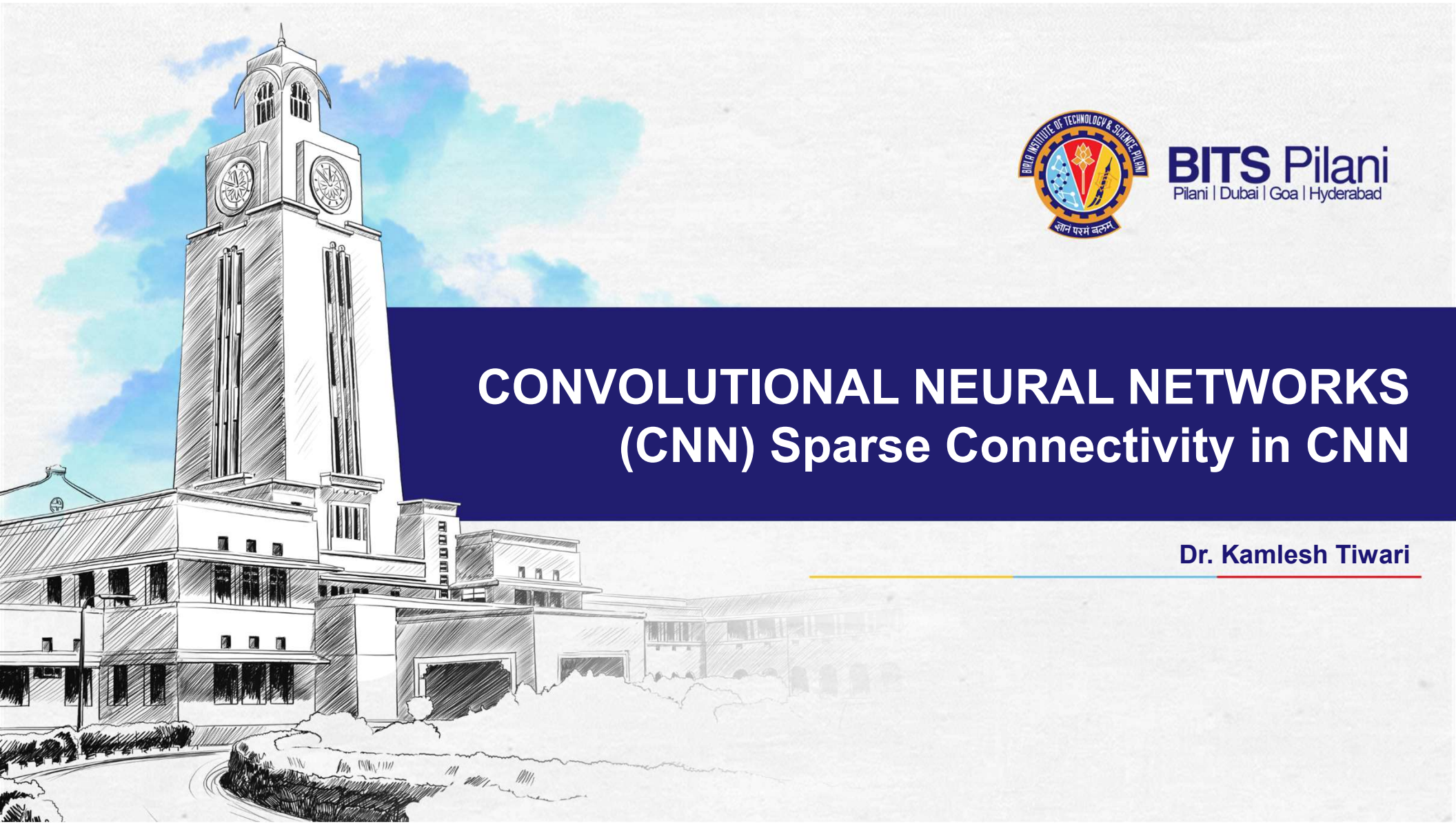
- Instead of handcrafted kernels, learn filters



- (Why not multiple?) learn multiple layers of kernels/filters



Treating these kernels as parameters and learning them.

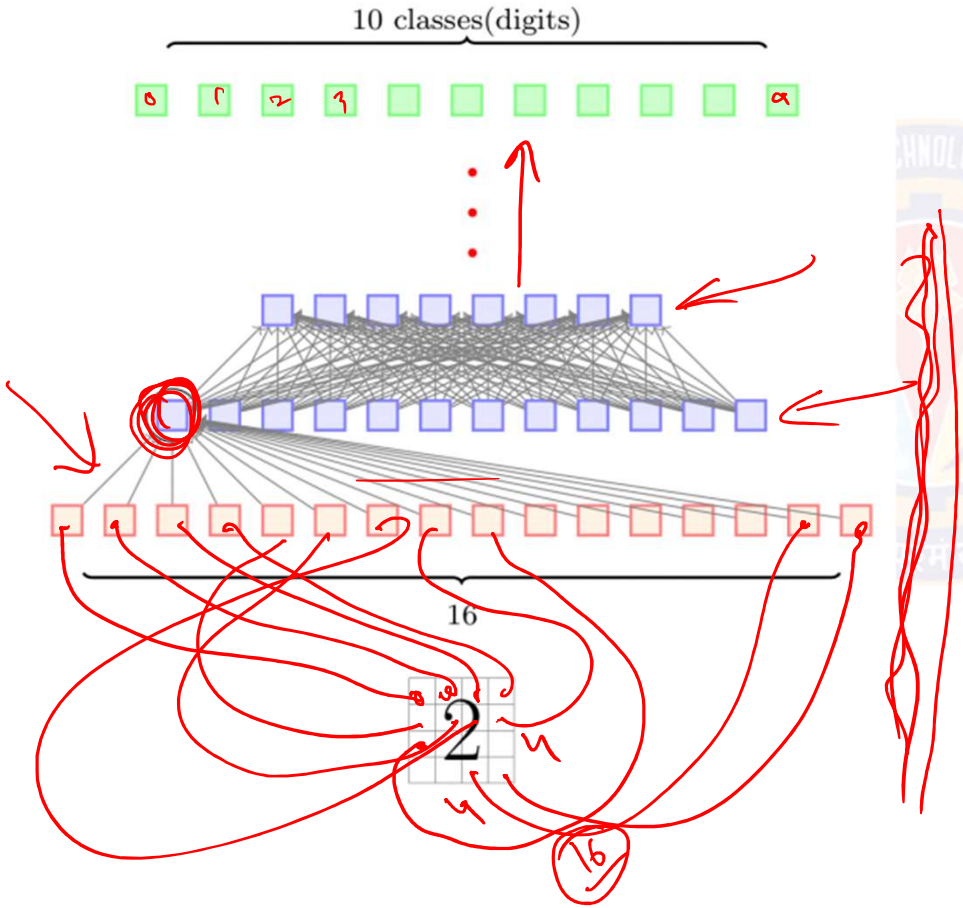


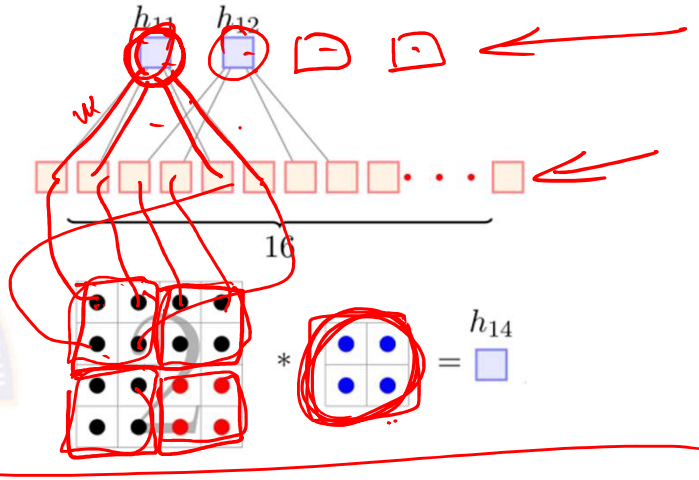
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS
(CNN) Sparse Connectivity in CNN

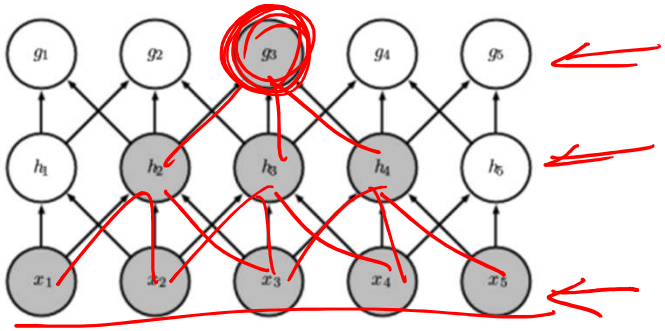
Dr. Kamlesh Tiwari

CNN has sparse connectivity with respect to NN



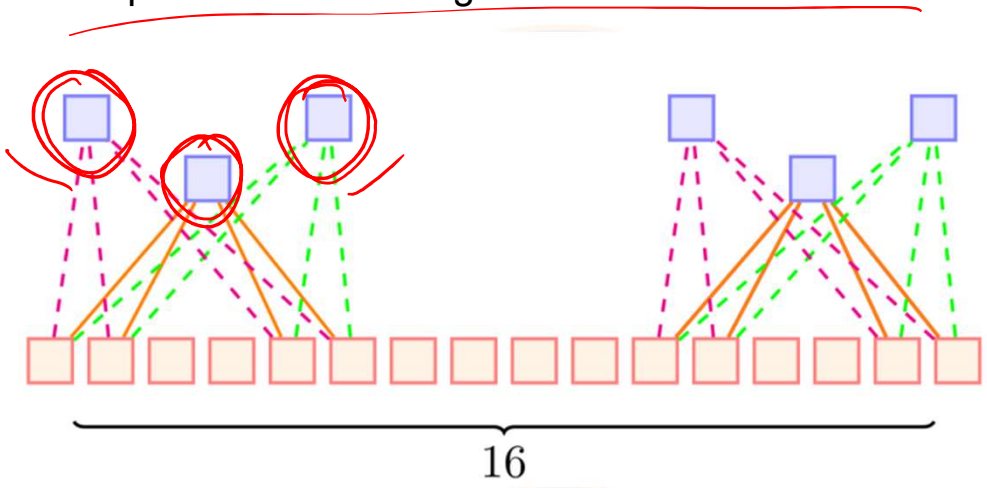


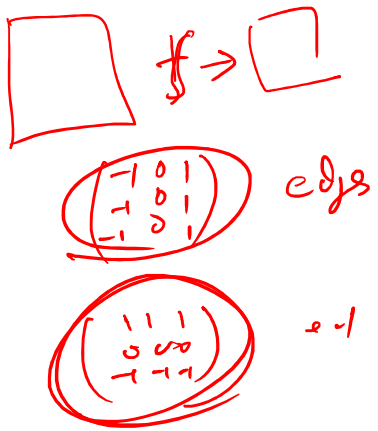
Interactions are preserved, even with reduced model parameters,



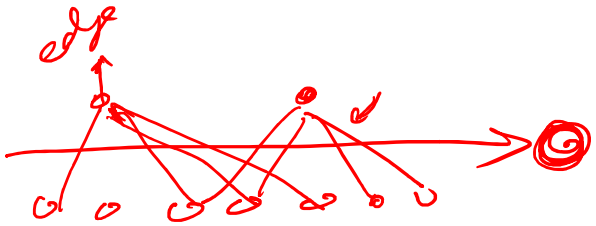
Weight sharing in CNN

- One place you are extracting edge other place something else? So we do not want the kernel to be different for different portions of the image.

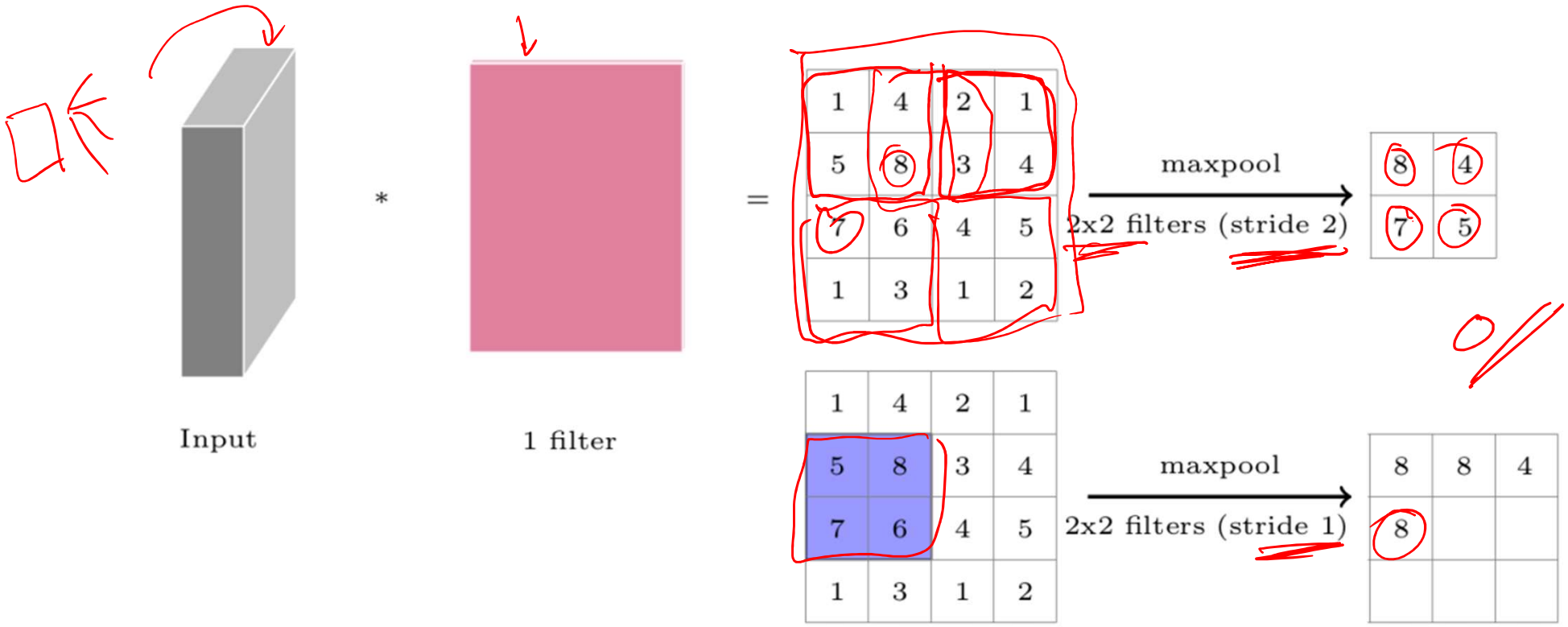




- Weight sharing in CNN makes the job of learning weights easier
- Multiple kernels help get different feature at the same level



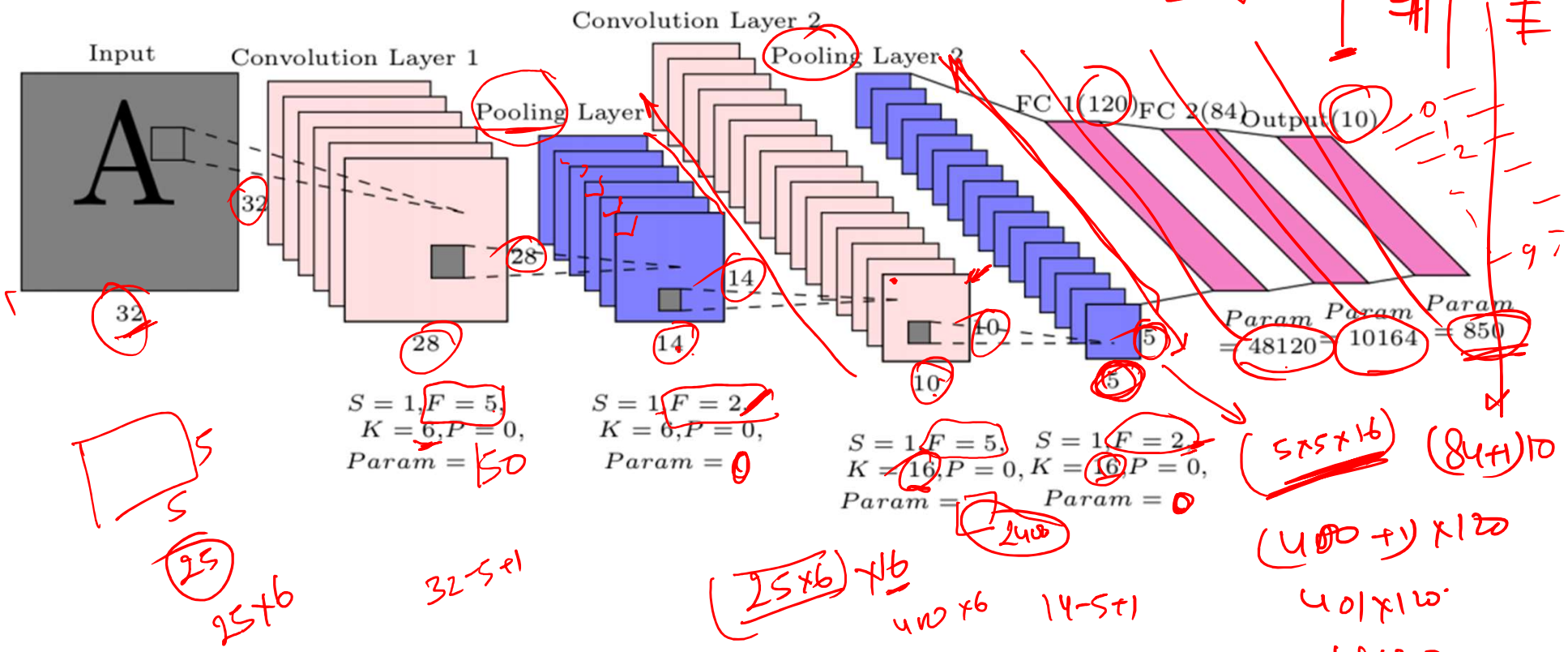
Pooling (max, min, average)



- Training CNN? backpropagation!

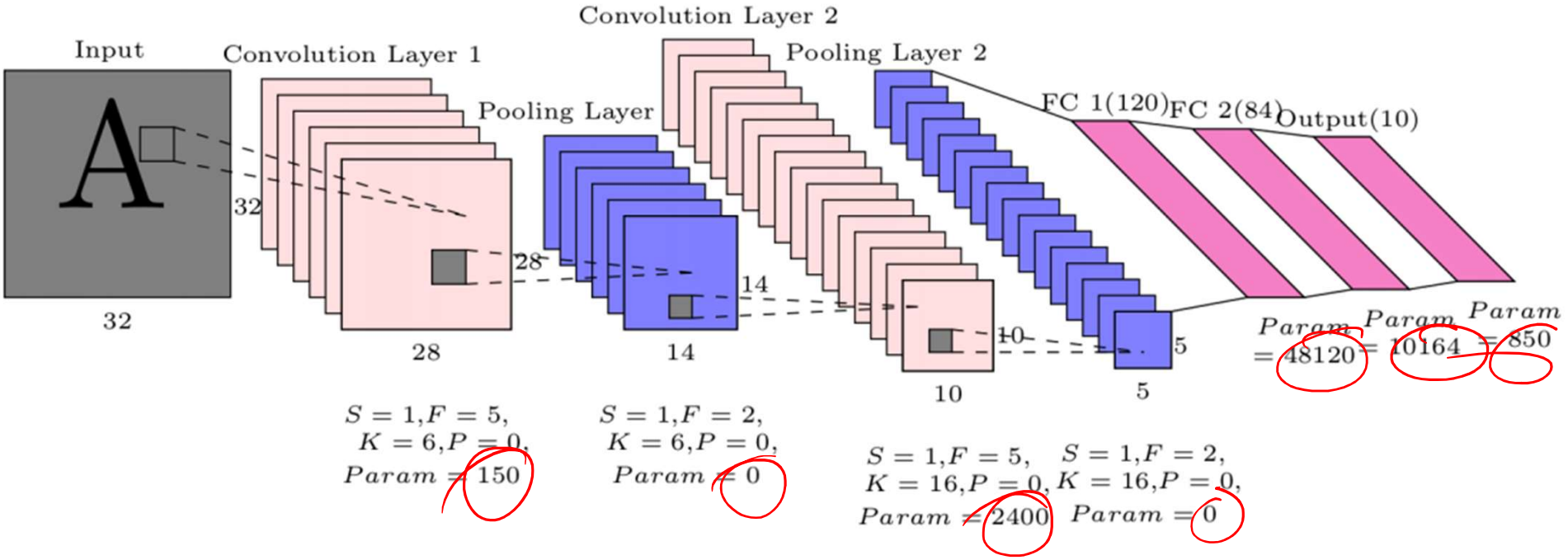
Case-Study: LeNet-5

Handwritten character recognition



Yann Lecun and Leon Bottou and Yoshua Bengio and Patrick Haffner, Gradient-based learning applied to document recognition, pp 2278–2324, IEEE-1998

LeNet-5 for handwritten character recognition

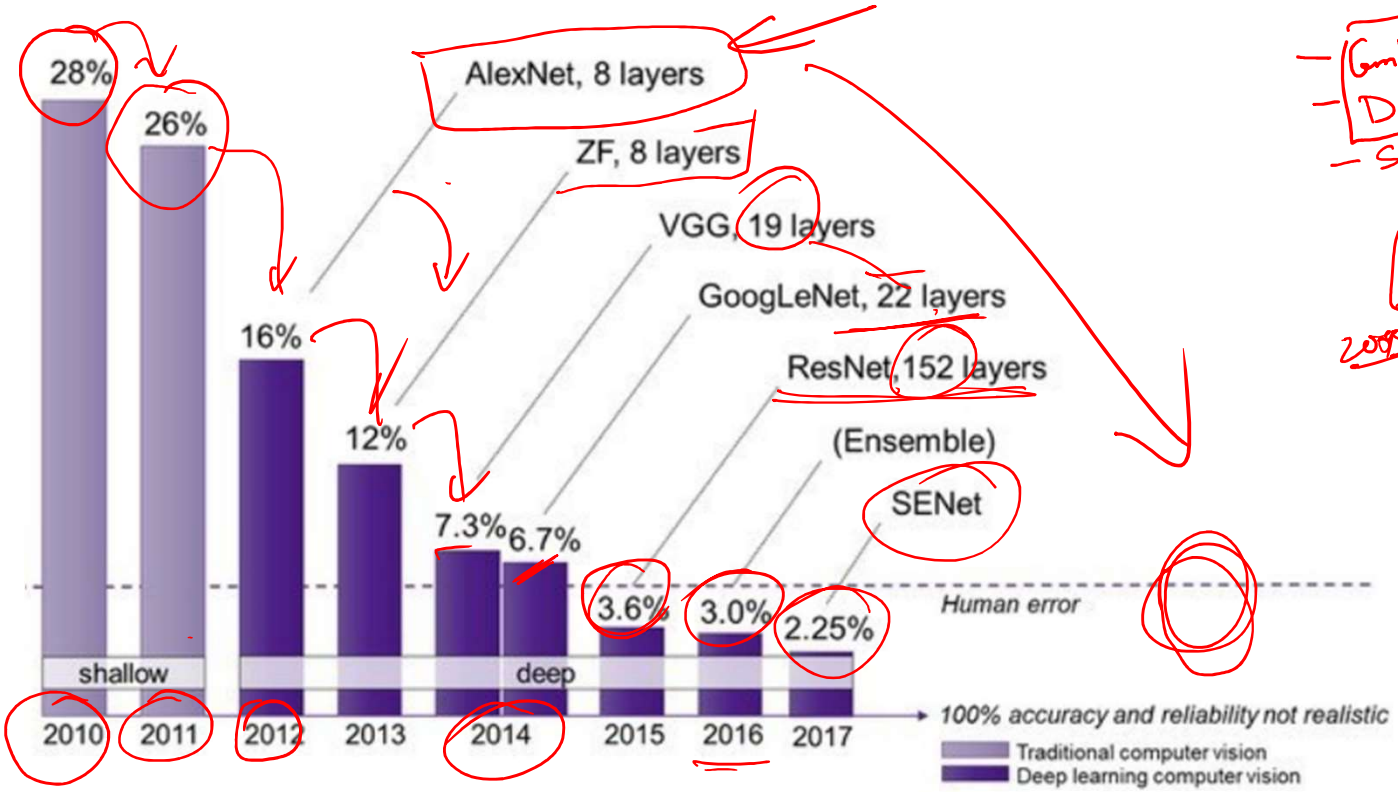


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS (CNN) Popular Architectures

Dr. Kamlesh Tiwari

ImageNet ILSVRC



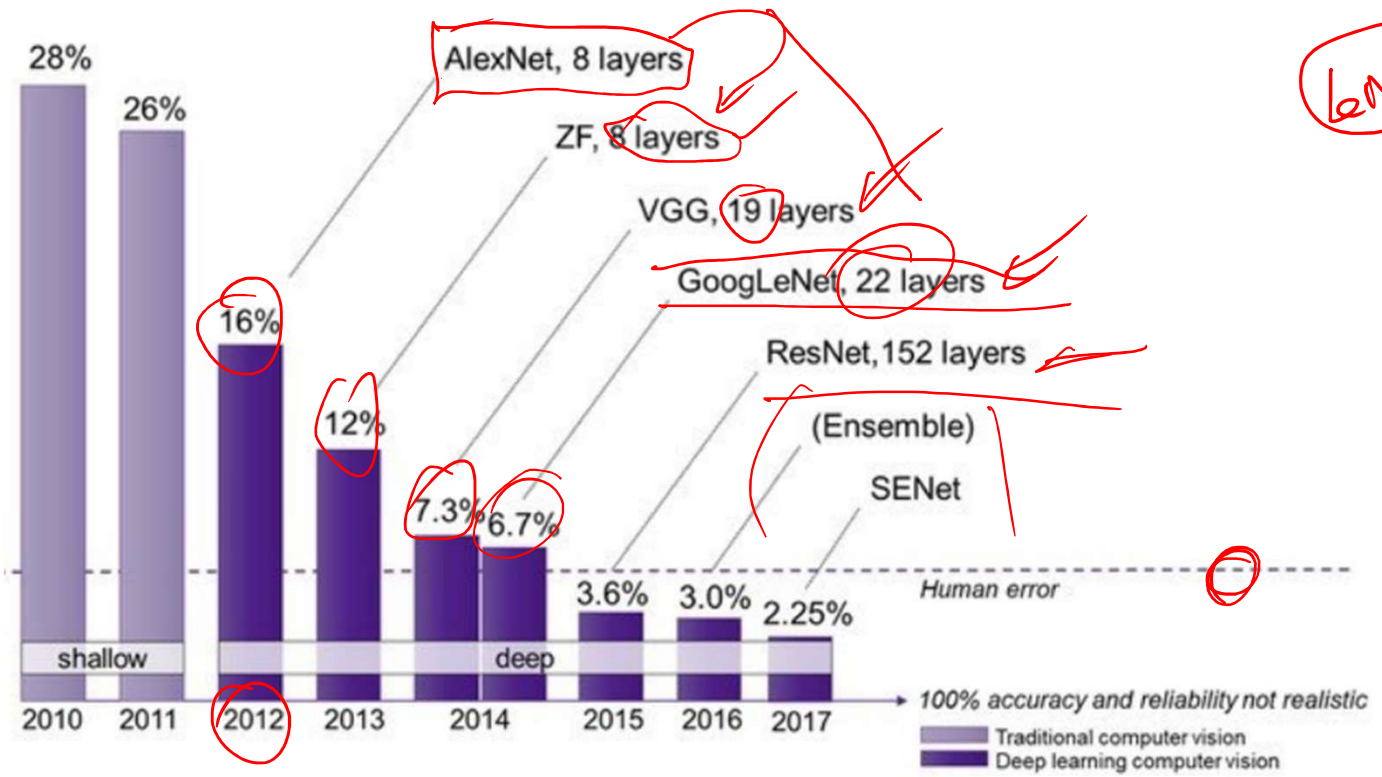
- (2009) 22K category, 14M images
- Challenge 1000 class, 1431167 images
- HoG, LBP, SVM ...

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS (CNN) AlexNET, ZF-NET, VGG-16

Dr. Kamlesh Tiwari

ImageNet ILSVRC



- (2009) 22K category, 14M images
- Challenge 1000 class, 1431167 images
- HoG, LBP, SVM ...

AlexNet

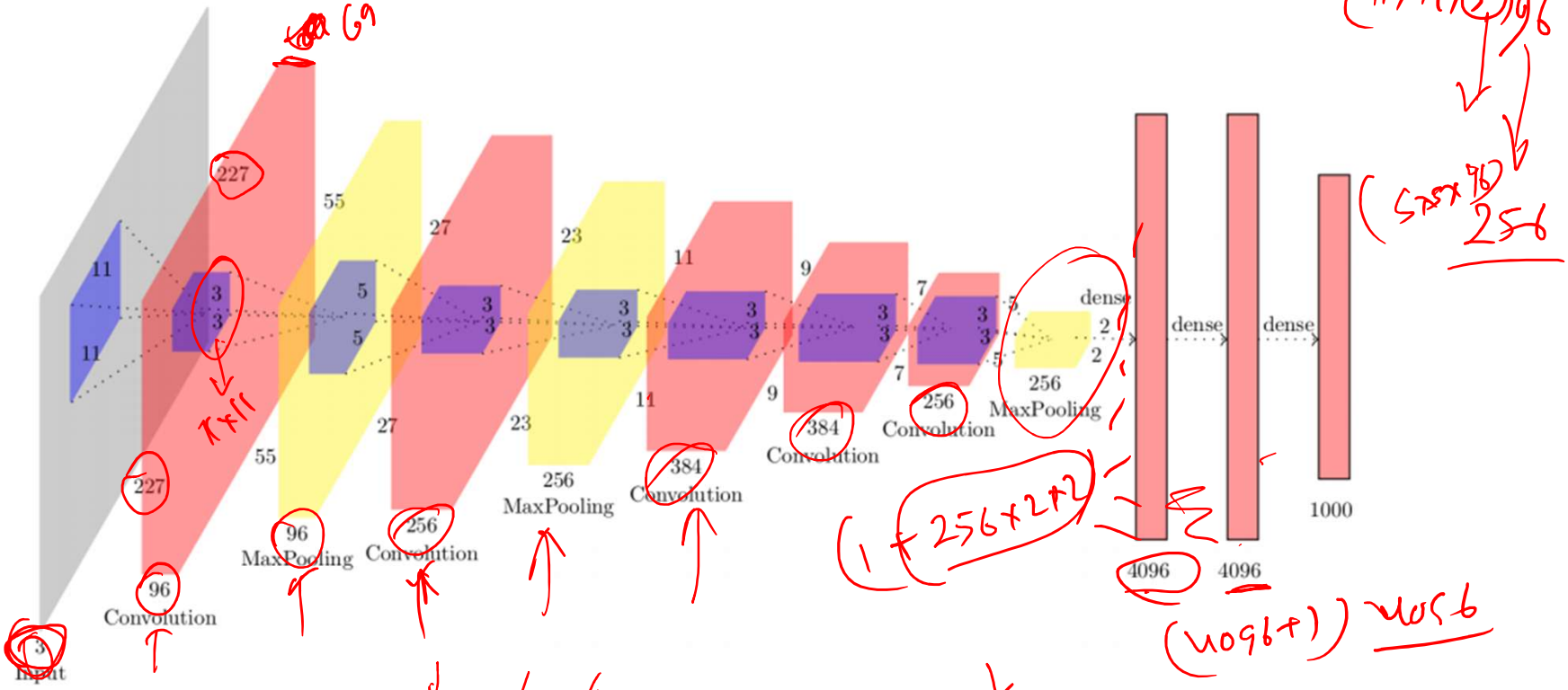
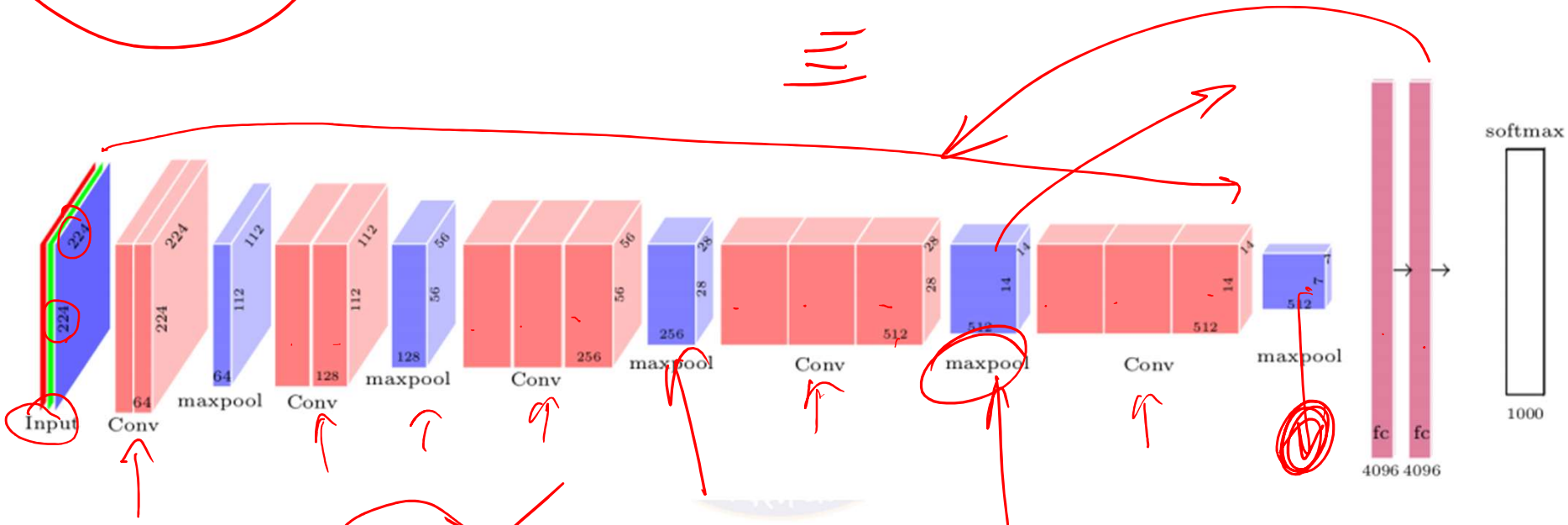


Image size 227 × 227 × 3 → [96 F=11, S=4, P=0] → [MaxPool F=3, S=2] → [256 F=5, S=1, P=0] → [MaxPool F=3, S=2] → [384 F=3, S=1, P=0] → [384 F=3, S=1, P=0] → [256 F=3, S=1, P=0] → [MaxPool F=3, S=2] → FC 4096 → FC 4096 → FC 1000

ZFNet

Image size 227 × 227 × 3 → [69 F=7, S=4, P=0] → [MaxPool F=3, S=2] → [256 F=5, S=1, P=0] → [MaxPool F=3, S=2] → [512 F=3, S=1, P=0] → [1024 F=3, S=1, P=0] → [512 F=3, S=1, P=0] → [MaxPool F=3, S=2] → FC 4096 → FC 4096 → FC 1000

VGG16



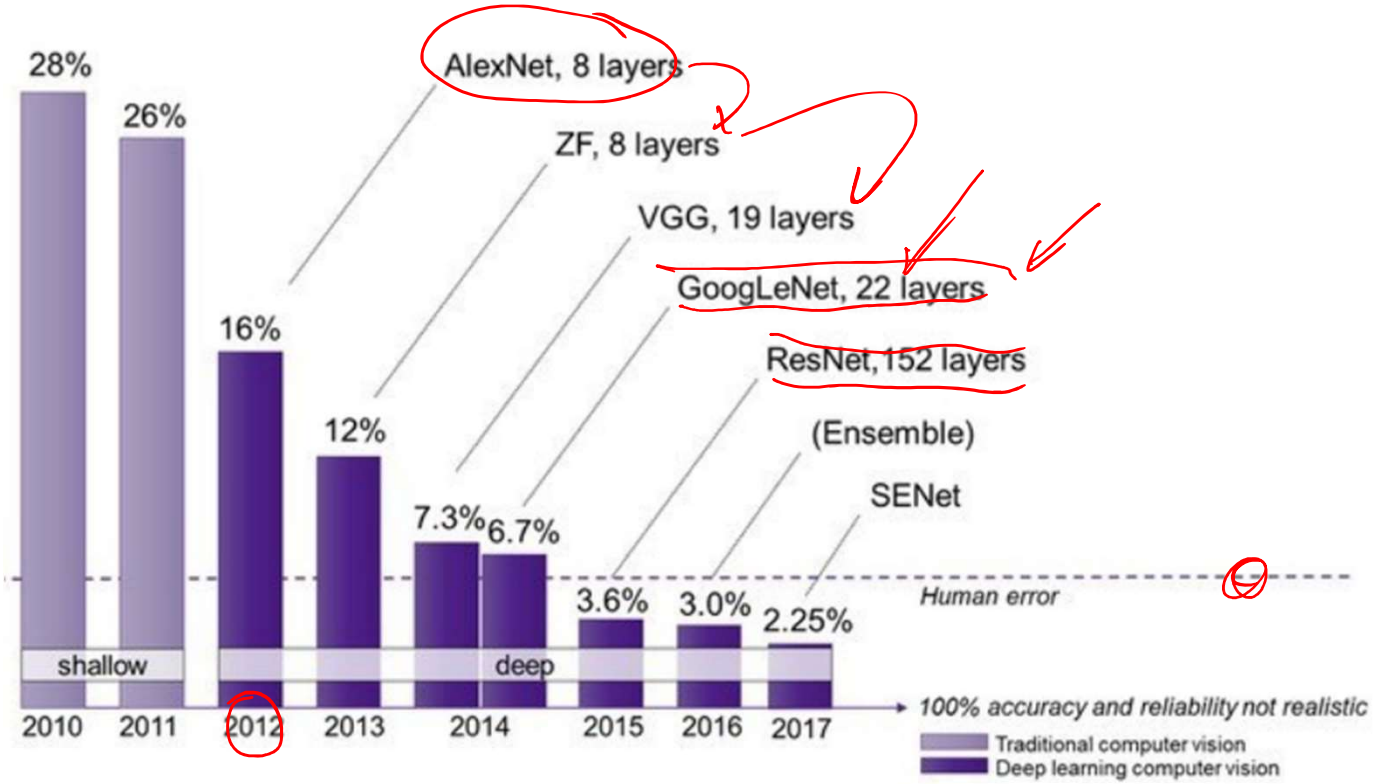
- Kernel size is always 3 × 3
- 16M parameters in pre-FC and 122 in FC. First FC layer is huge
- Layers represents abstract representation and can be reused (FC or Conv)

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CONVOLUTIONAL NEURAL NETWORKS (CNN) Inception, ResNET

Dr. Kamlesh Tiwari

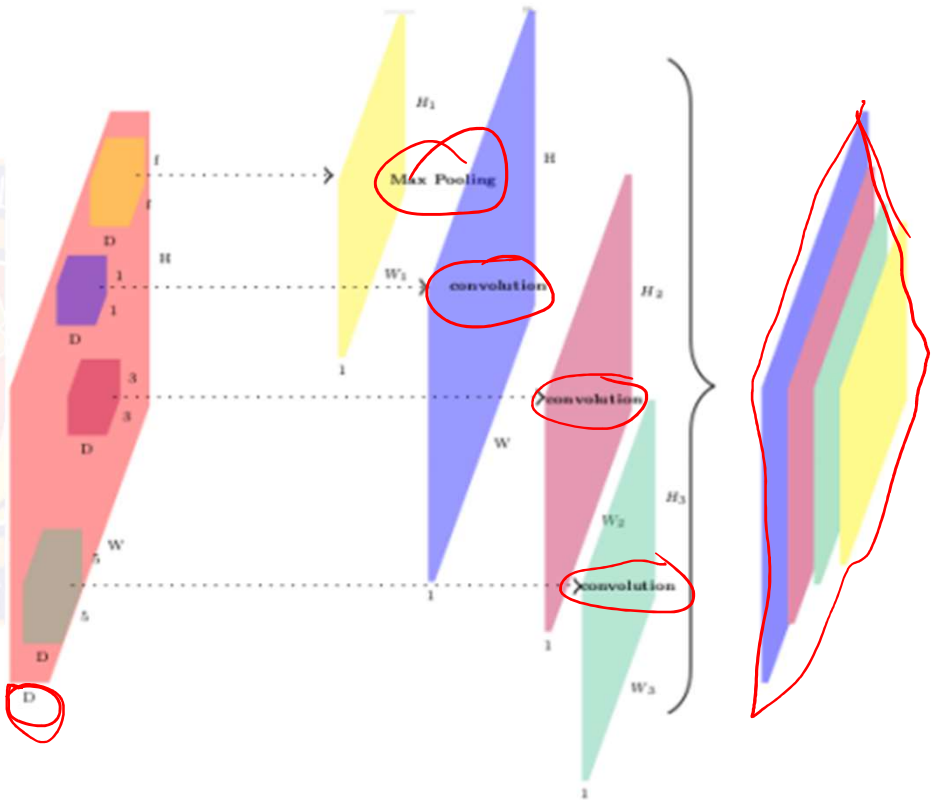
ImageNet ILSVRC



- (2009) 22K category, 14M images
- Challenge 1000 class, 1431167 images
- HoG, LBP, SVM ...

GoogLeNet

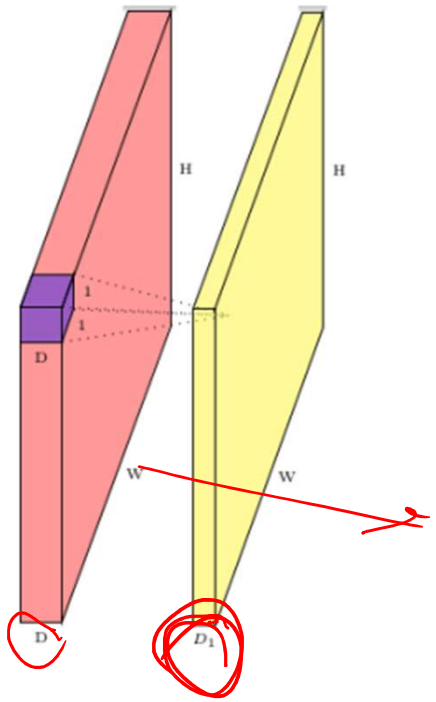
- Recall scale invariance in SIFT
- Multiple filters of different size is a good idea
- With W × H × D input and F × F × D
- Filter and S = 1 and no padding, output is of size (W − F + 1) × (H − F + 1)
- Each value needs F × F × D computation



Can we reduce this computation a bit?

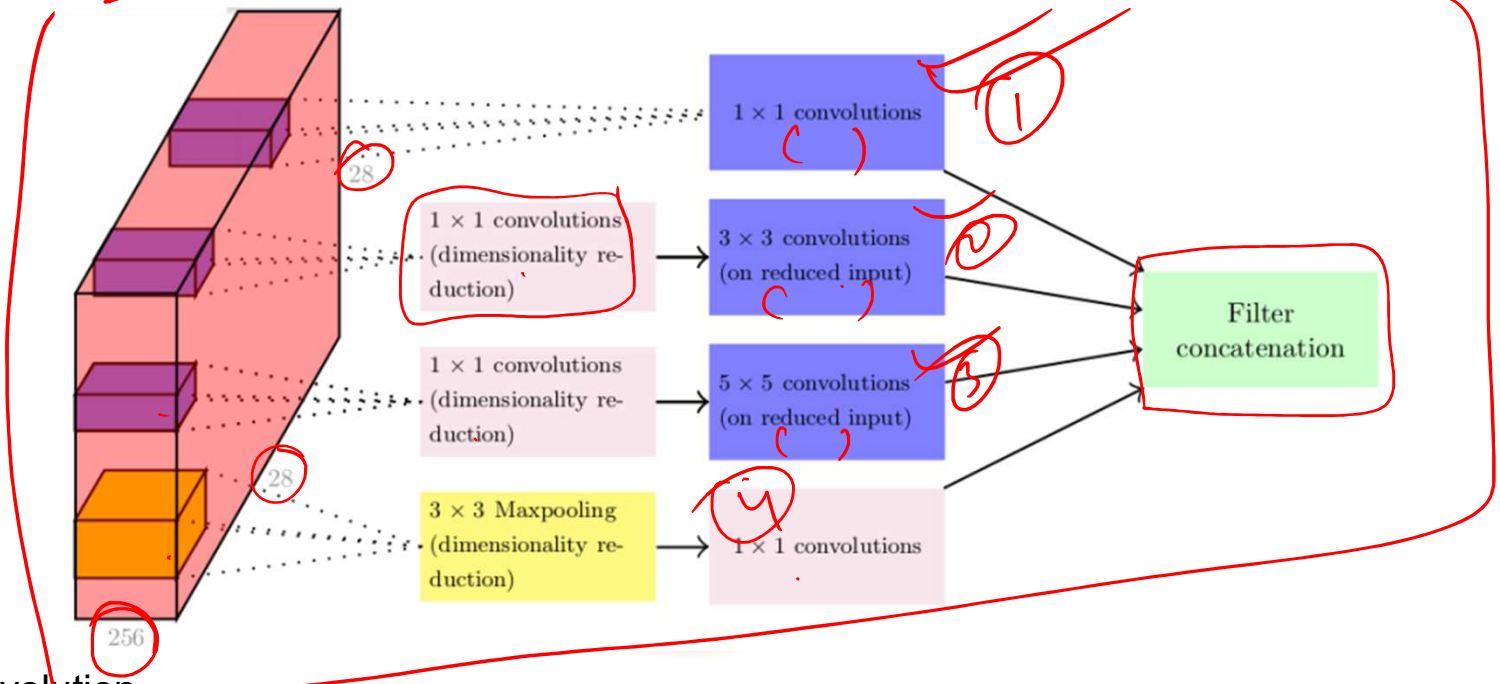
- Idea is to have 1 × 1 computation

1 × 1 convolution



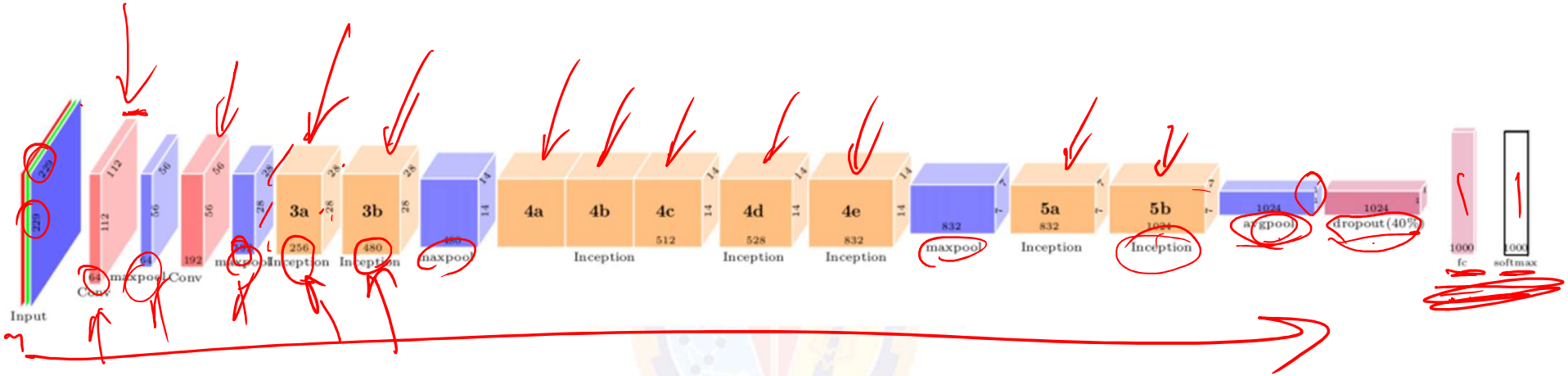
- 1 × 1 is 1 × 1 × D
- They produce one output place
- By using D1 such 1 × 1 convolution output becomes F × F × D1
- We have D1 < D

Inception Block: Multiple convolutions



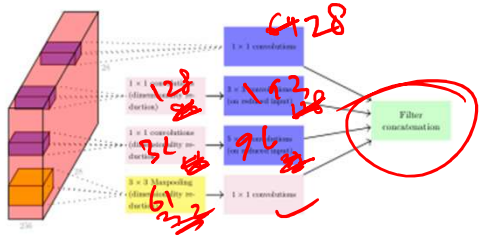
- 1 × 1 convolution
- 1 × 1 convolution followed by 3 × 3
- 1 × 1 convolution followed by 5 × 5
- 3 × 3 maxpool followed by 1 × 1
- Appropriate padding is done to make things of same size

GoogLeNet

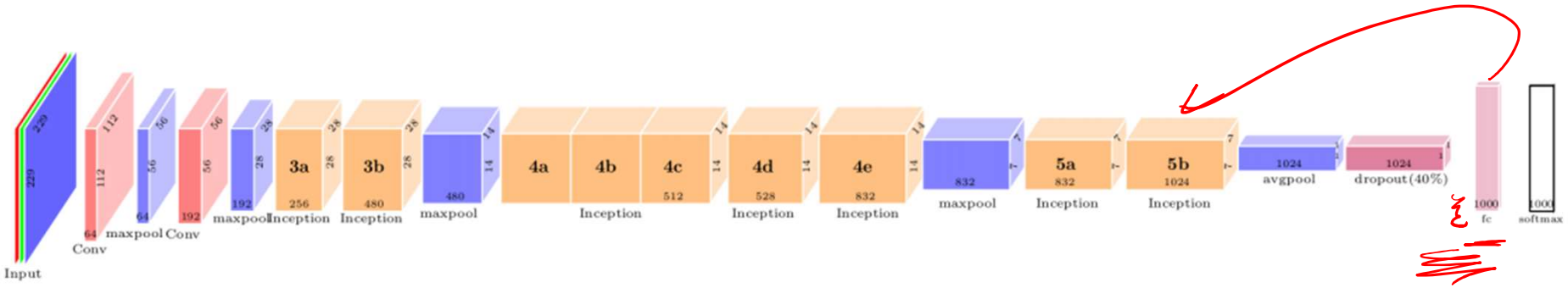


- Input is RGB 229 × 229
- Each inception module have very specific configuration.

(3a)	192 × 28 × 28	64 96 128 16 32 32
(3b)	256 × 28 × 28	128 128 192 32 96 64
(4a)	480 × 14 × 14	192 96 208 16 48 64
(4b)	512 × 14 × 14	160 112 224 24 64 64
(4c)	512 × 14 × 14	128 128 256 24 64 64
(4d)	512 × 14 × 14	112 144 228 32 64 64
(4e)	528 × 14 × 14	256 160 320 32 128 128
(5a)	832 × 7 × 7	256 160 320 32 128 128
(5b)	832 × 7 × 7	384 192 384 48 124 128



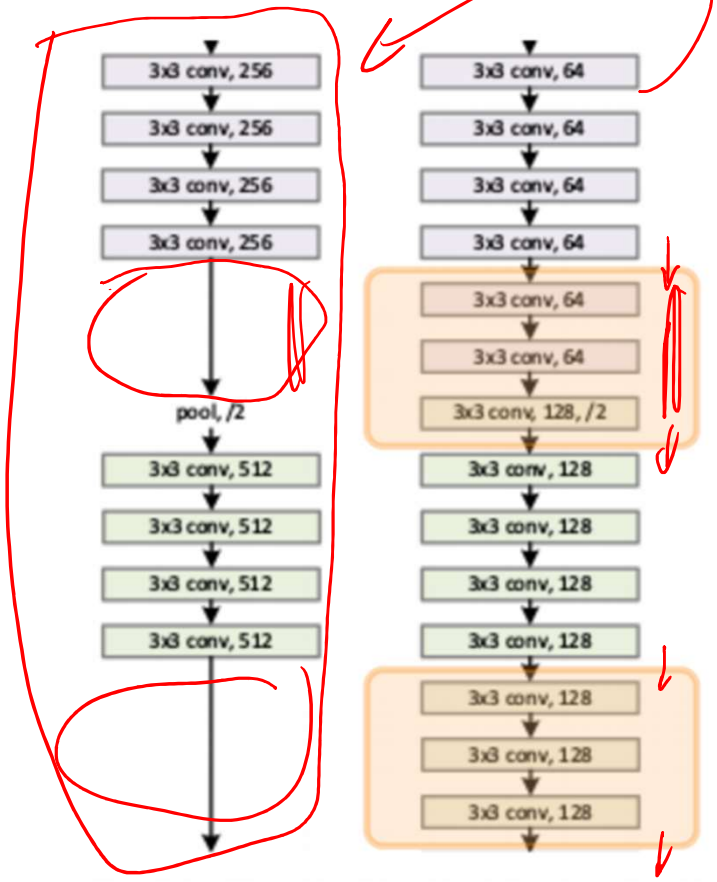
GoogLeNet



- VGGNET has 512 × 7 × 7 size at pre-FC this was an issue to connect with 4096
- GoogLeNet applies a average pool. Gives 49 time reduction. has 1024 values only
- Dropout and connect to 1000
- 12 times less connections as compared to AlexNet
- 2 times more computation as compared to AlexNet
- Very high accuracy. Error reduced from 16% -to- 6.7%

ResNet

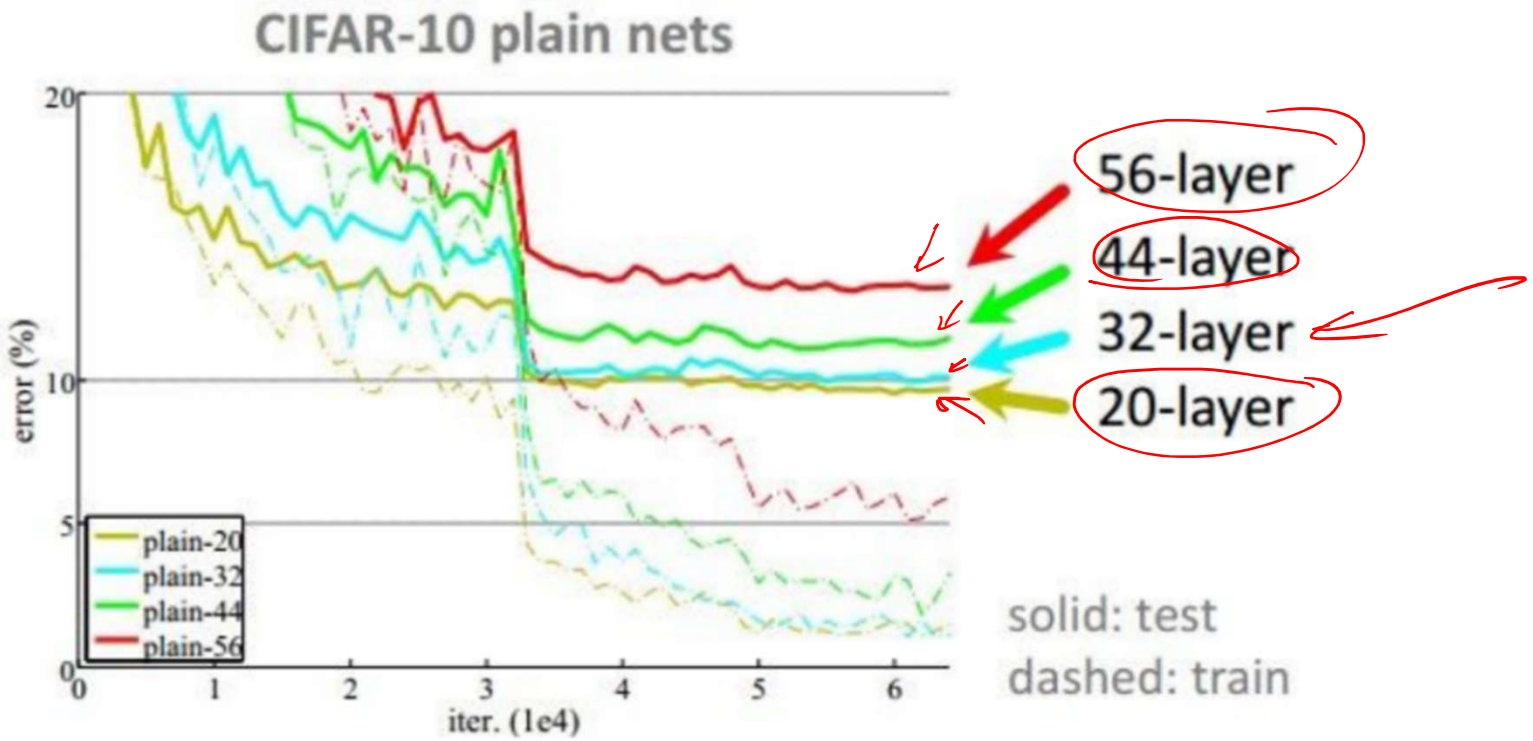
If a shallow neural network works well. What would happen if we add more layers?



- Deep network should also work well (It would learn identity in new layers)

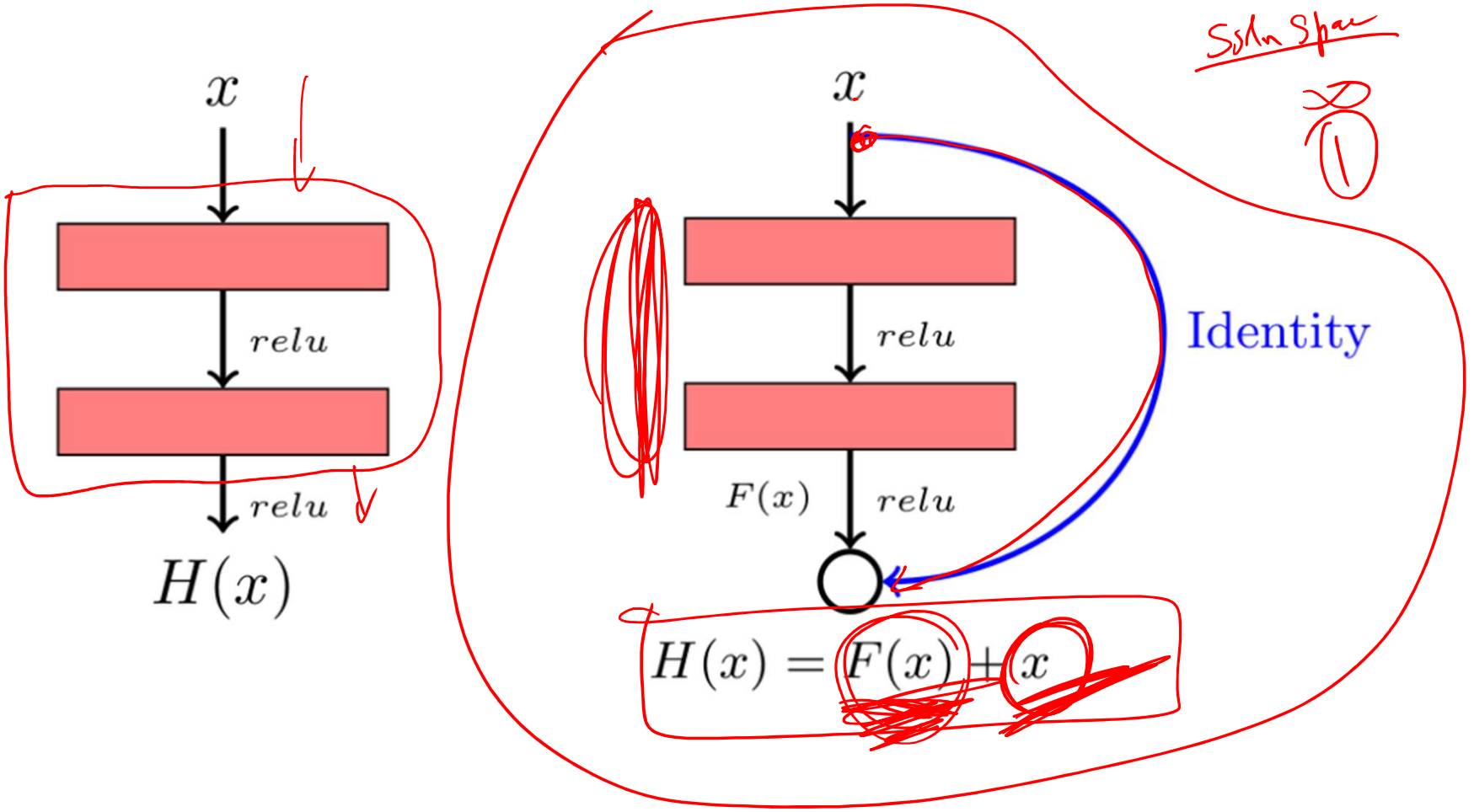
ResNet

But, in practice it was not happening

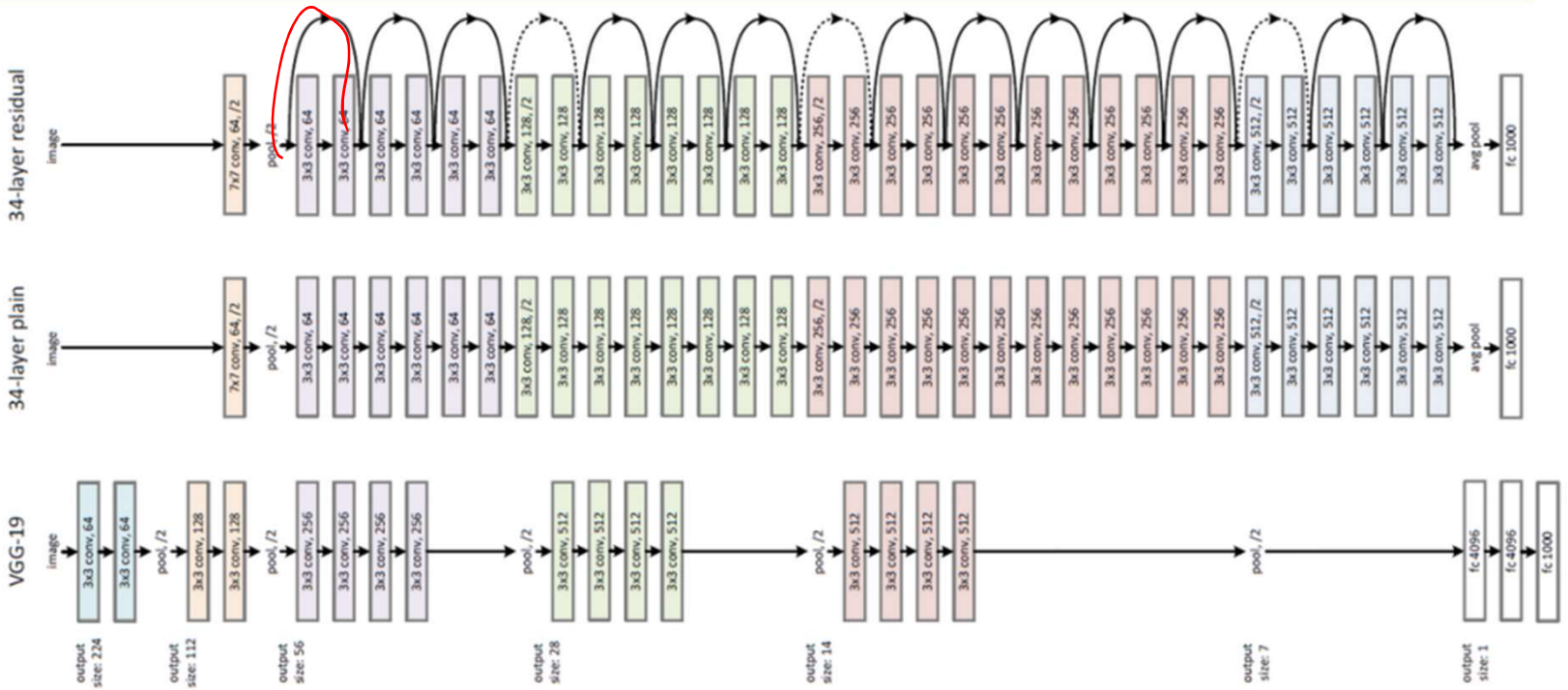


Why? Identity is one of the solution in large domain.

Let me tell this to the network



ResNet Comparison



34-layer ResNet with Skip / Shortcut Connection (Top), 34-layer Plain Network (Middle), 19-layer VGG-19 (Bottom)

152-layer deep net was better than human. Only 3.6% error rate ImageNet Classification

Better than the 2nd best system ImageNet Detection: 16% ImageNet Localization: 27% COCO Detection: 11% COCO Segmentation: 12%

ResNet Hyper-parameters and Issues

- Training takes huge time
- Batch Normalization
- Zavier/2 initialization
- SGD and momentum
- Small learning rate 0.1
- Mini-batch size 256
- Weight decay
- No Dropout

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to Deep Learning

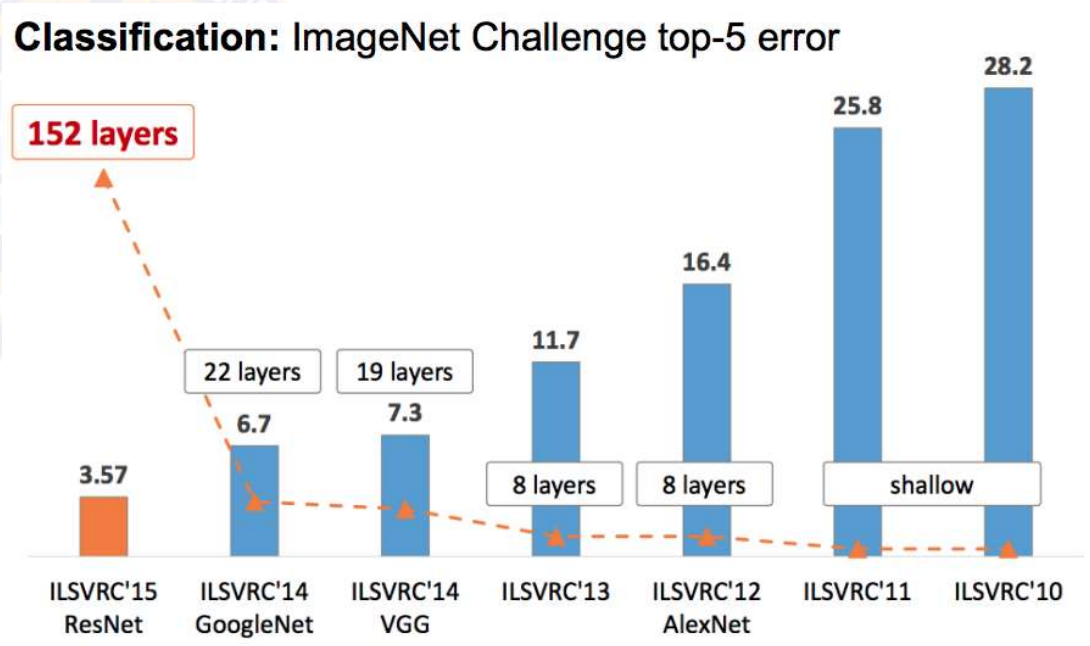
Kamlesh Tiwari

Deep Learning (DL)

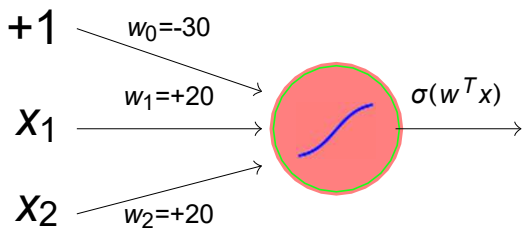
Deep neural networks are capable of solving very complex problems

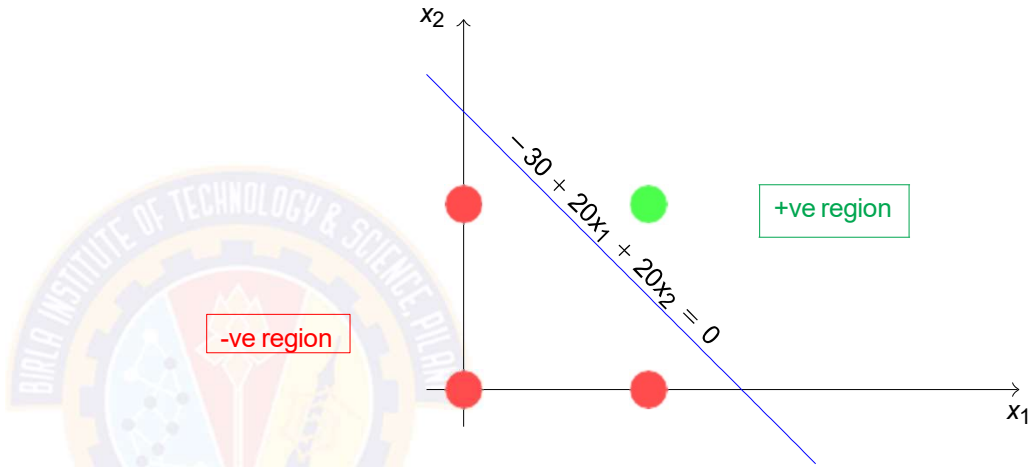
- ImageNet Challenge (2011): identify objects in images

  (4 million images in 21841 categories [Challenge 1000 categories])

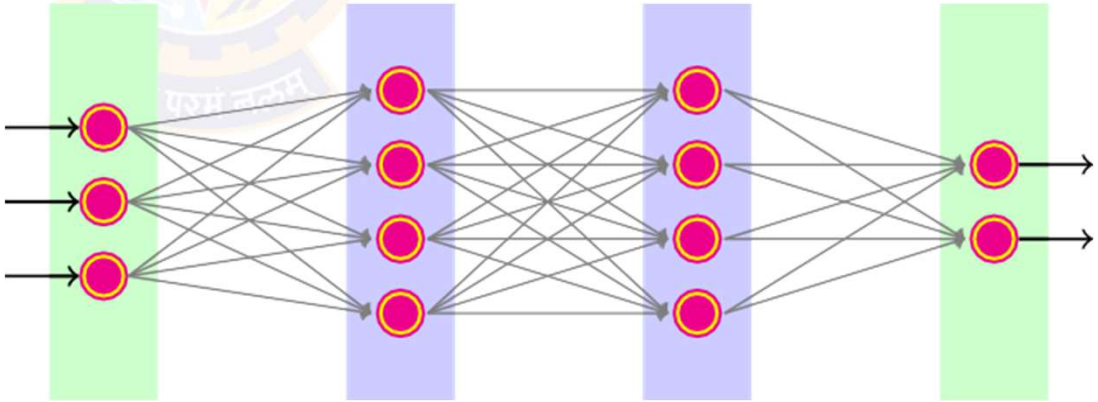


Essentially it Represents A Decision Boundary

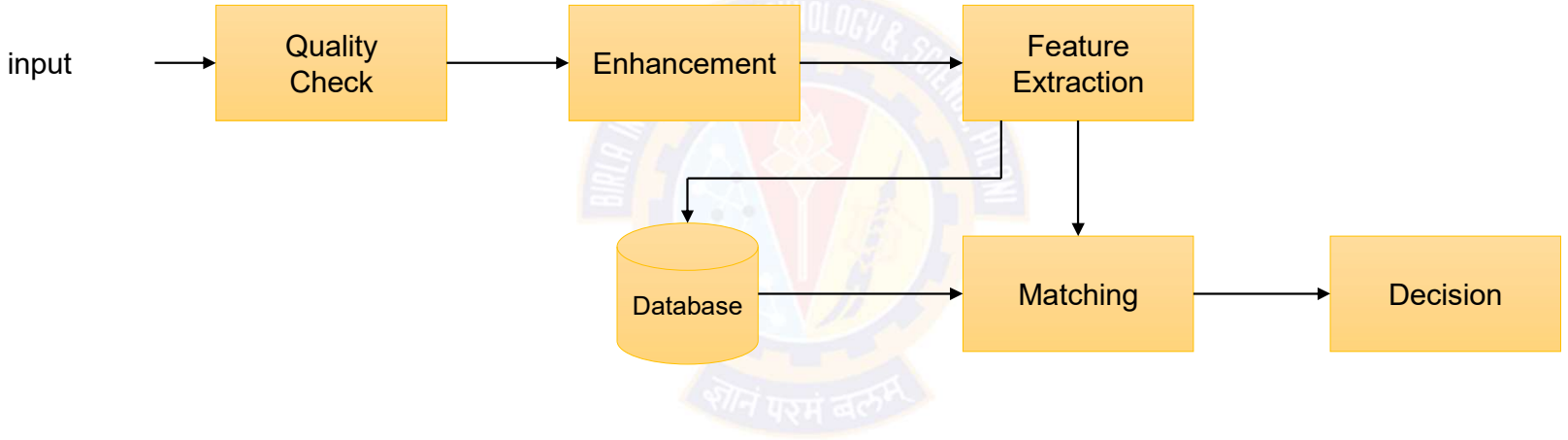




However, complex boundaries could be made using multiple neurons and stacking them in a layer.

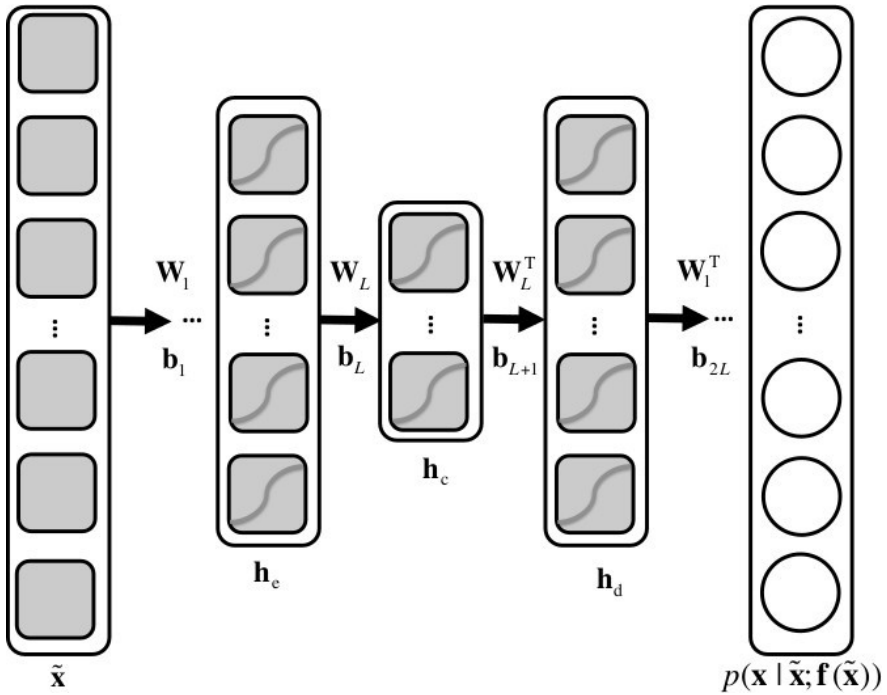


General ML Pipeline



Deep Learning can provided end-to-end learning

Autoencoders



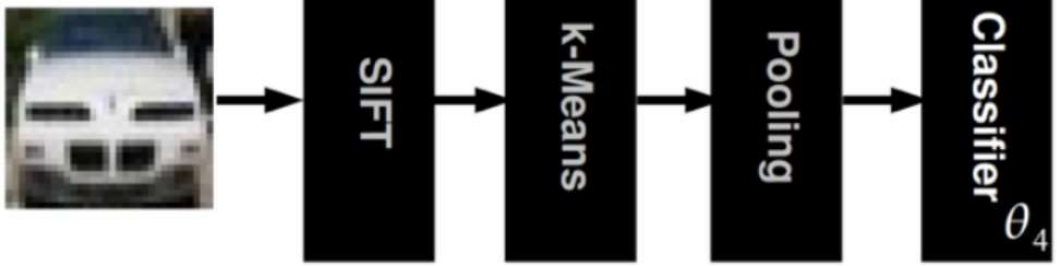
- An unsupervised learning algorithm, setting the target values to be equal to the inputs
- Learns an approximation

Deep Learning

- **Shallow to Deep:** is linear combination to composition

$$\sum_i g_i \quad \rightarrow \quad g_1(g_2(g_3(...)))$$

- Why?



- Optimization is difficult for a non-convex, non-linear systems
- Freezing some layers makes it shallow.

Key Points

- Traditional way to hand engineer feature is brittle and not scalable
- **Why now?** because we have data (imageNet), compute power (GPUs) and software (tensorFlow)
- DL learns underlying features/representations directly from data.
- DL have been found useful to many real life problems. Capable of solving very complex problems.
- As the data increases algorithm DL becomes better
- Deep learning is NOT just adding more layers to a neural network.

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DEEP LEARNING AND ANN

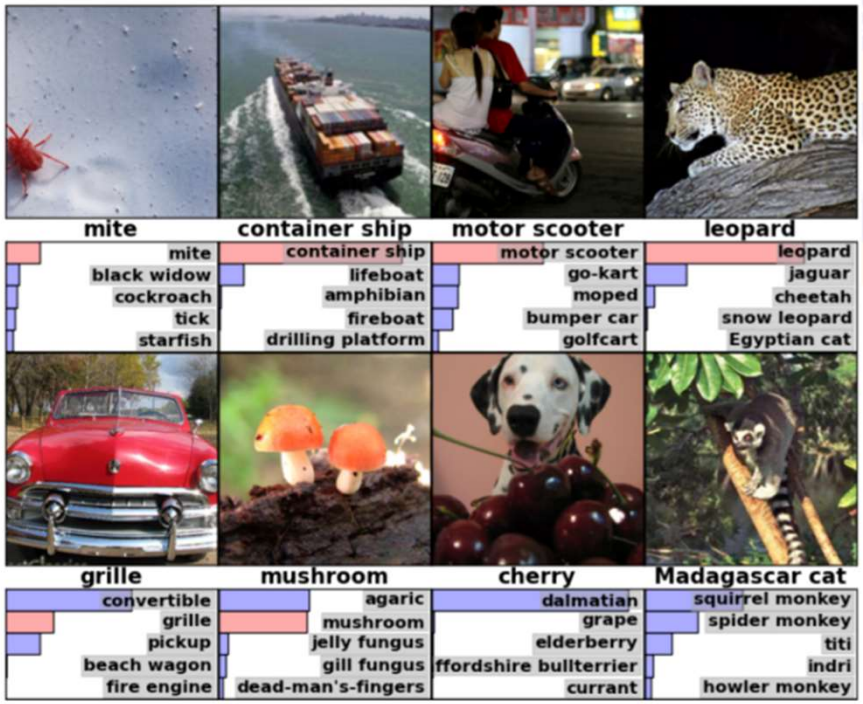
Dr. Kamlesh Tiwari

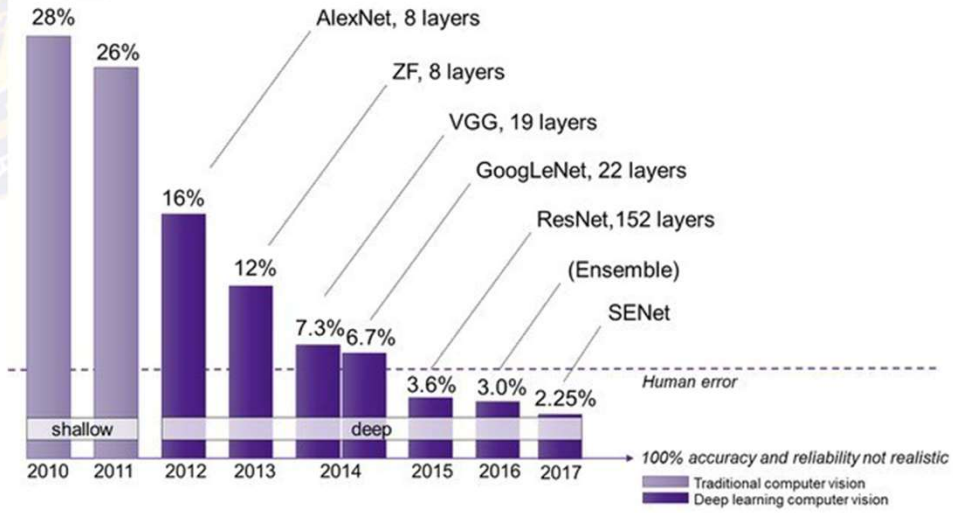
Deep Learning (DL)

An approach to learn data using neural network

Capable of solving very complex problems

- ImageNet Challenge (2011): identify objects, 4 million images in 21841 categories (Challenge 1000 categories)





Deep Learning (DL)

An approach to learn data using neural network

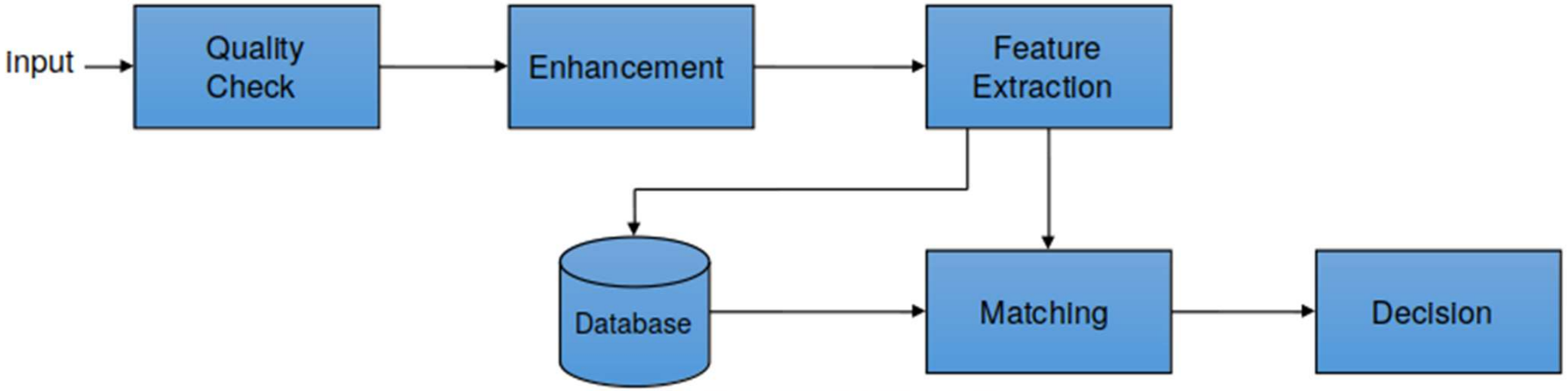
Capable of solving very complex problems

- ImageNet Challenge (2011): identify objects in images (millions)
- Generate audio: temporal dependence

DL learns underlying features/representations directly from data

- Traditional way to hand engineer feature is brittle and not scalable
- DL involves NO low, mid and high level feature pipeline
- Why now? because we have data (imageNet), compute power
- (GPUs) and software (tensorFlow)
- Deep learning is NOT just adding more layers to a neural network

General ML Pipeline



Various choices are there at all levels

DL provides end-to-end learning

Deep Learning

- Shallow to Deep: is linear combination to composition

$$\sum_i g_i \quad \rightarrow \quad g_1(g_2(g_3(\ldots)))$$

- Why

- Optimization is difficult for a non-convex, non-linear systems
- Freezing some layers makes it shallow

Thank You!

Next Session: DL

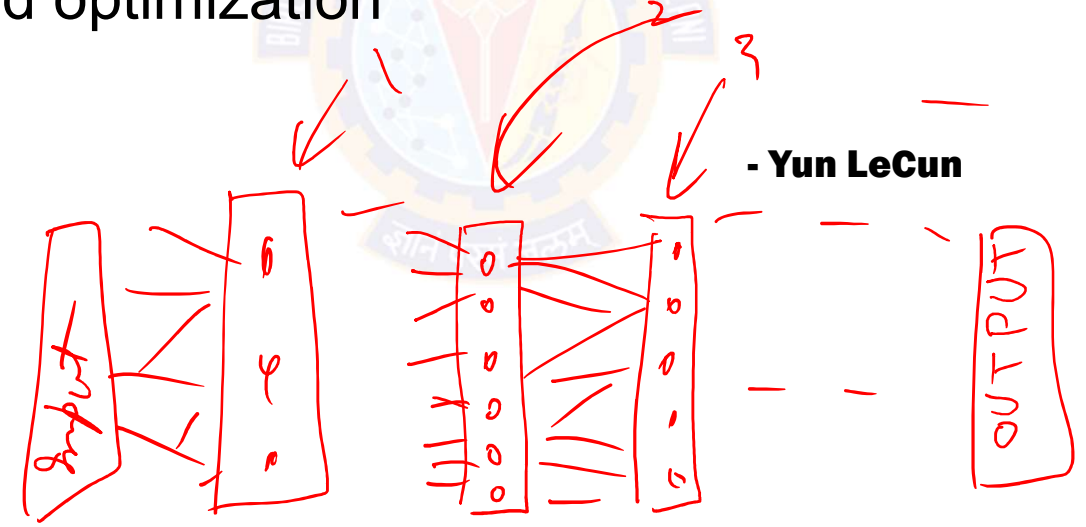
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DEEP LEARNING AND ANN

Dr. Kamlesh Tiwari

Deep Learning

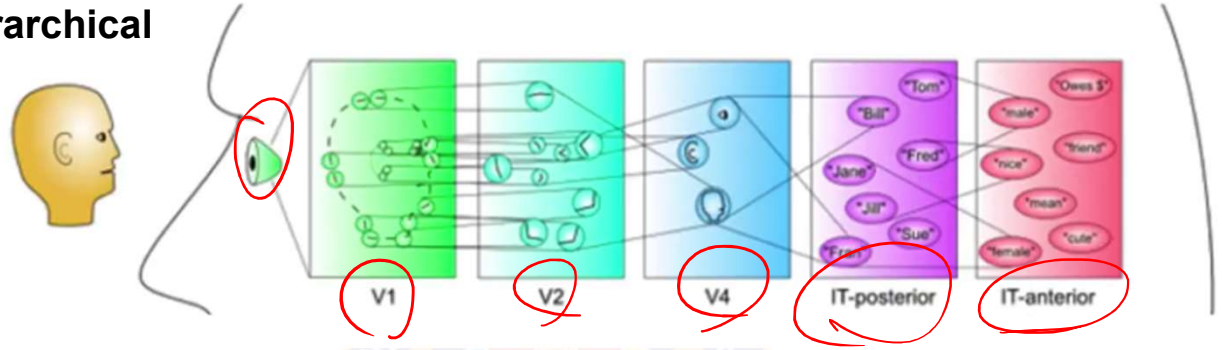
DL is constructing network of parameterized functional modules and training them from examples using gradient based optimization



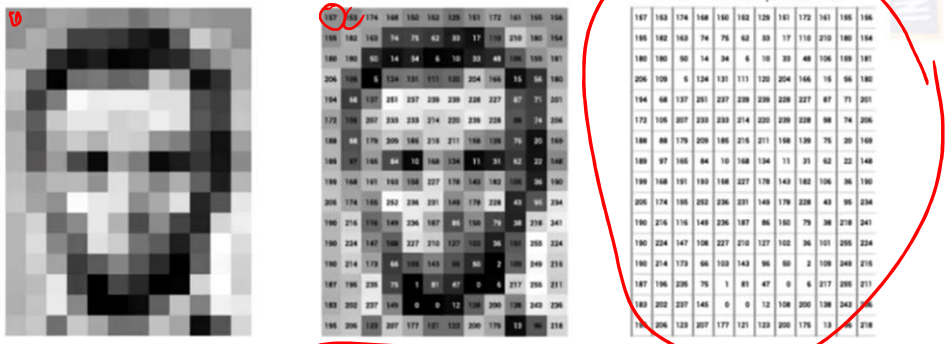
- Yun LeCun

Visual Cortex is Hierarchical

Visual Cortex is Hierarchical



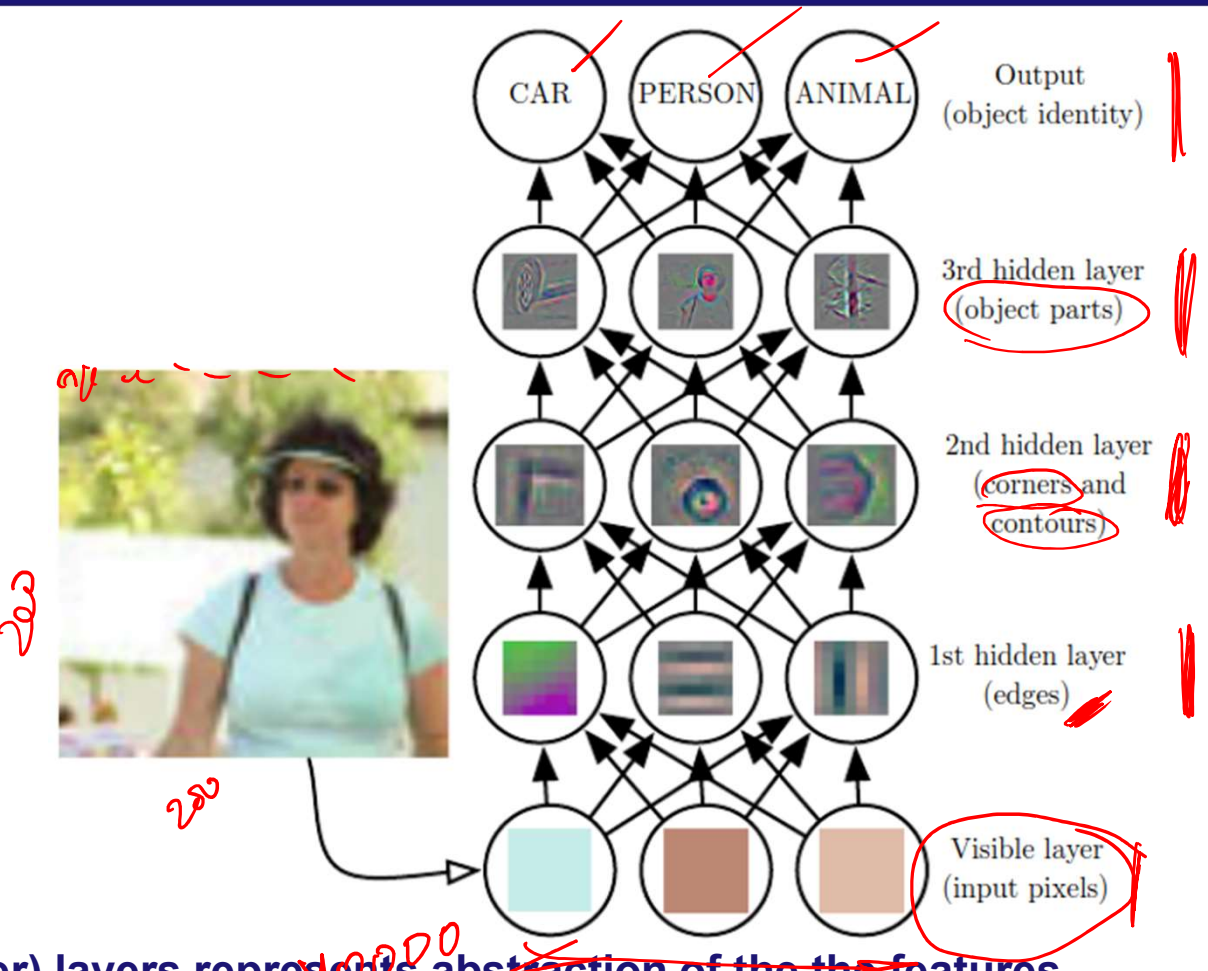
Images are numbers (so determine feature)



There could be occlusion, viewpoint variation, scale variation, illumination variation, deformation, background clutter, noise, intra-class variation and much more.

DL helps to get hierarchy of features

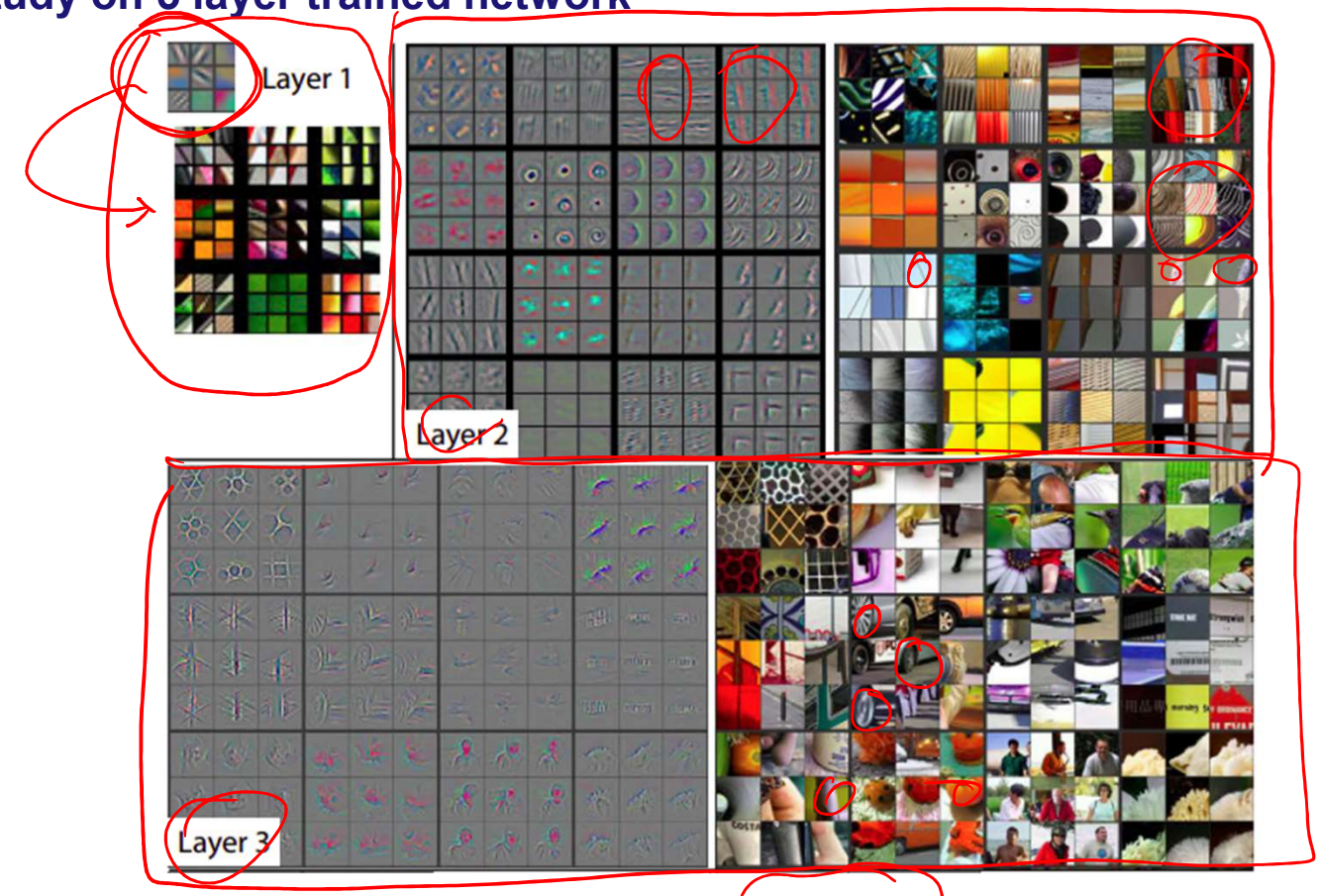
DL Builds Hierarchy of Features



Higher (or deeper) layers represents abstraction of the the features

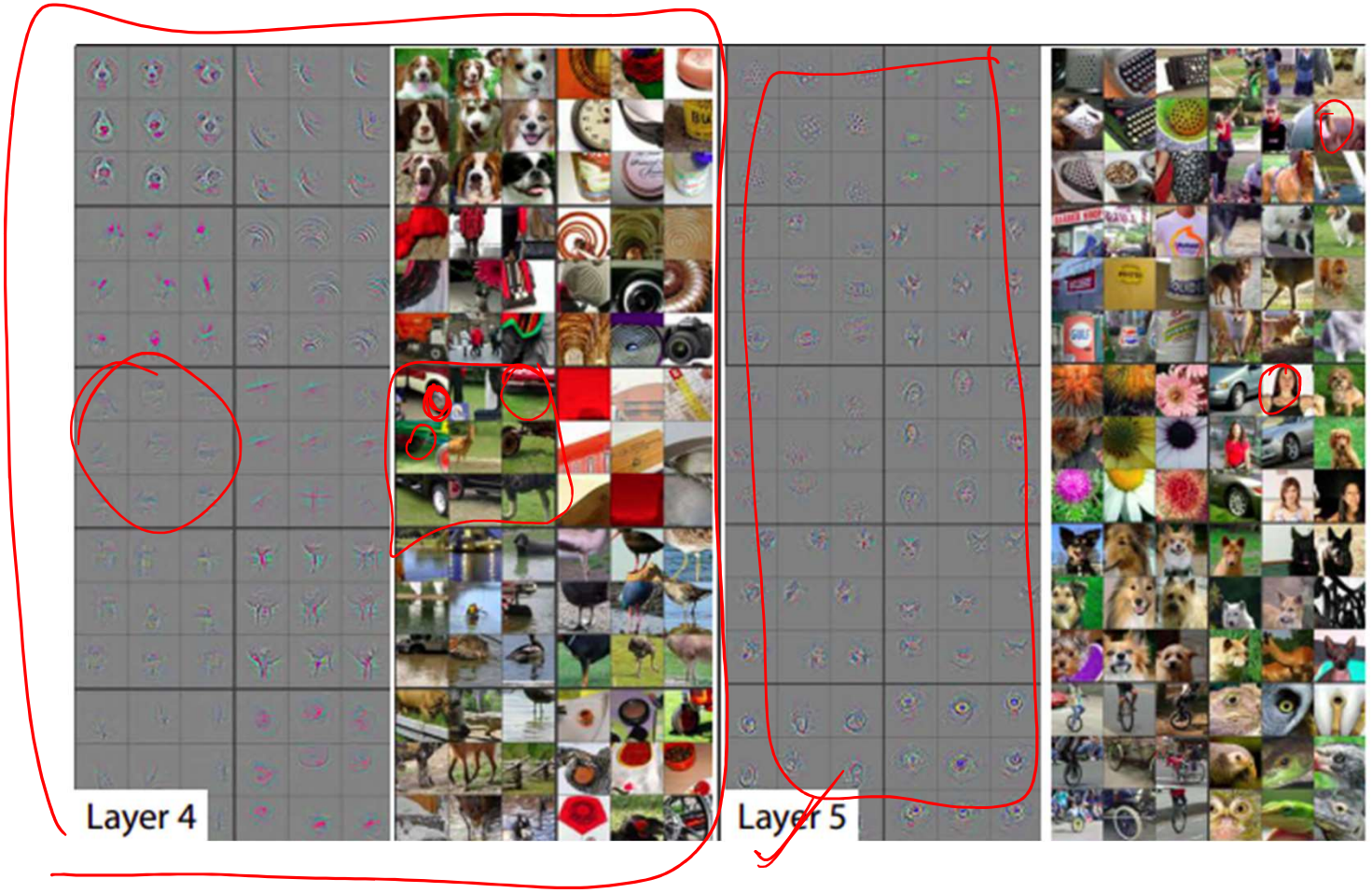
ECCV2014 - Visualizing and Understanding CNN

A case study on 5 layer trained network



Visualizing and Understanding Convolutional Networks, ECCV2014

ECCV2014 - Visualizing and Understanding CNN



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DEEP LEARNING AND ANN (hyperparameters)

Dr. Kamlesh Tiwari

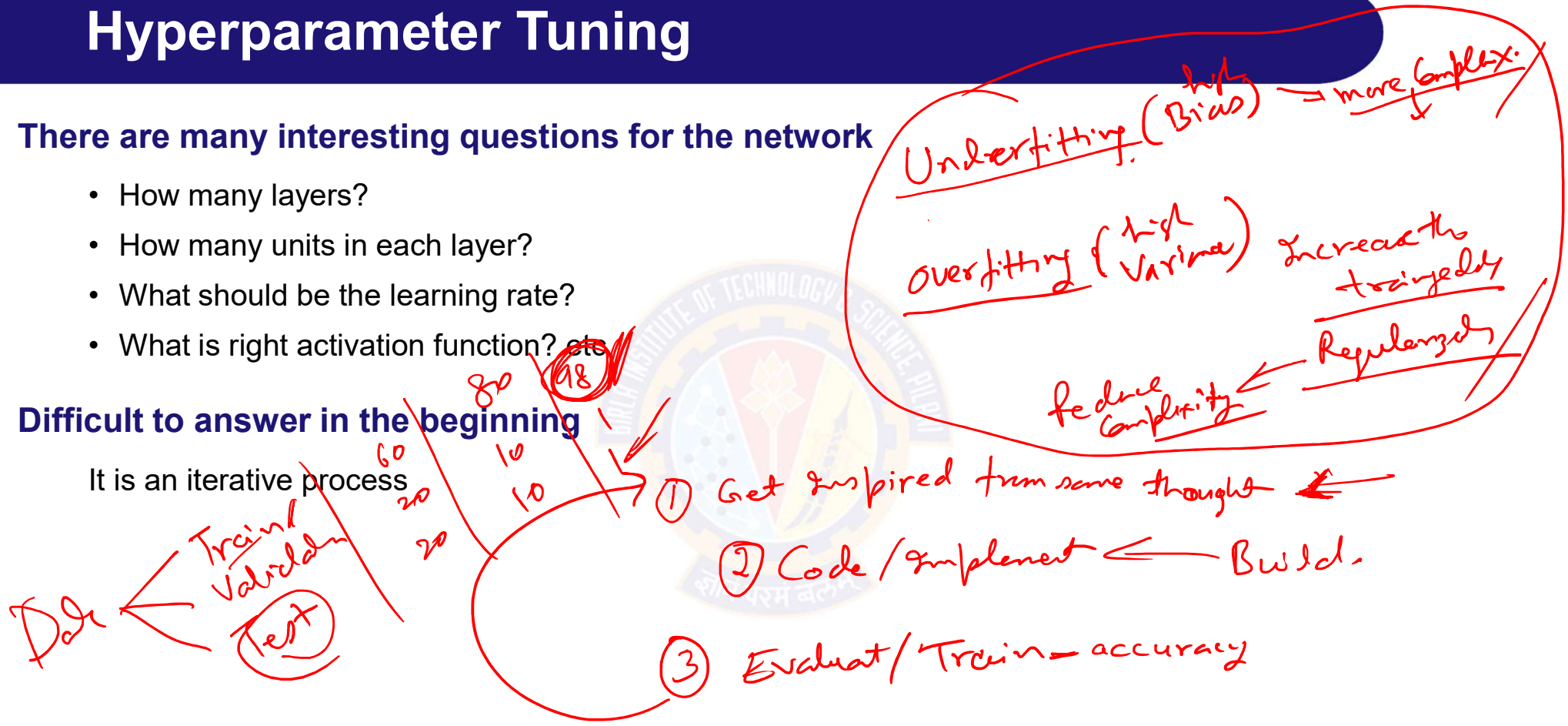
Hyperparameter Tuning

There are many interesting questions for the network

- How many layers?
- How many units in each layer?
- What should be the learning rate?
- What is right activation function? etc

Difficult to answer in the beginning

It is an iterative process

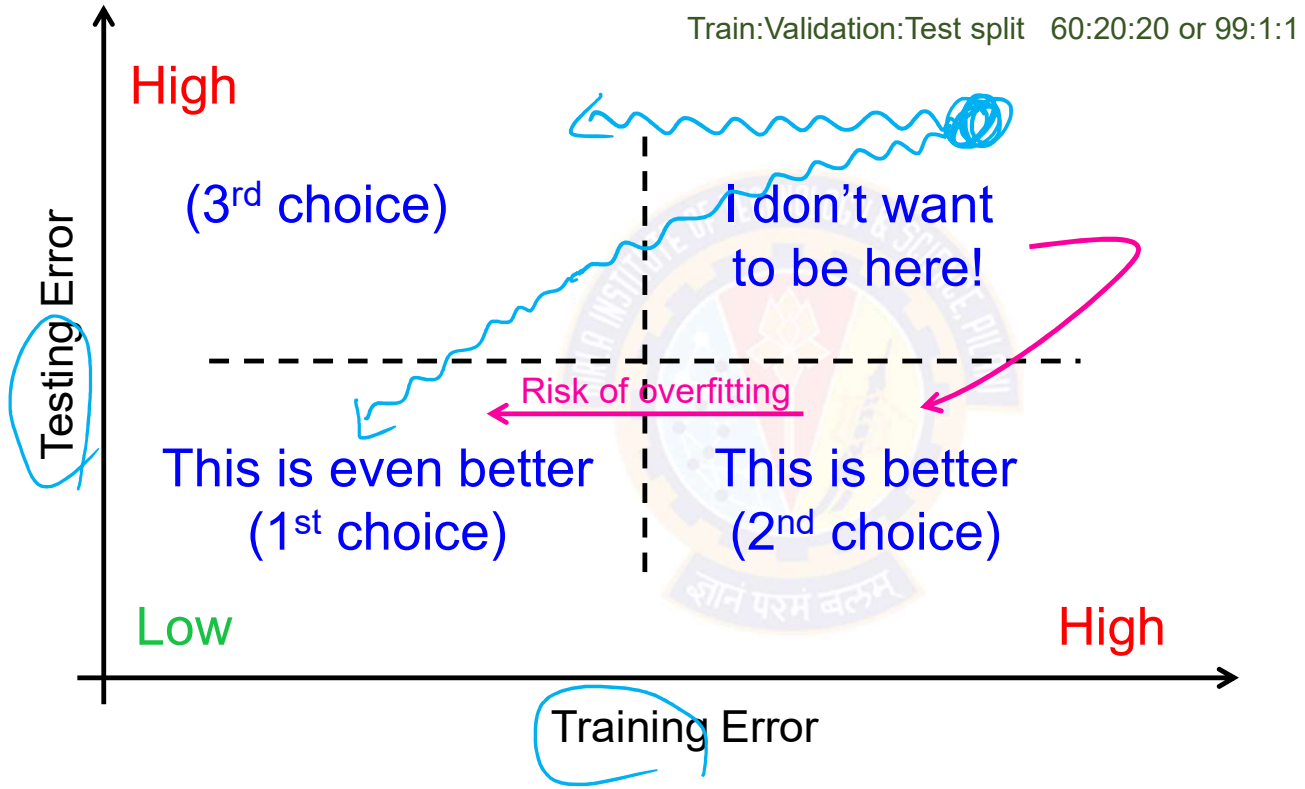


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DEEP LEARNING AND ANN

Dr. Kamlesh Tiwari

Which side do you want to be?



Low training error comes with a risk of overfitting (high variance)

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DEEP LEARNING AND ANN
Weight Decay

Dr. Kamlesh Tiwari

Regularization for logistic regression

- Optimization minimize loss $min_w\ J(w)$ by adjusting w where

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

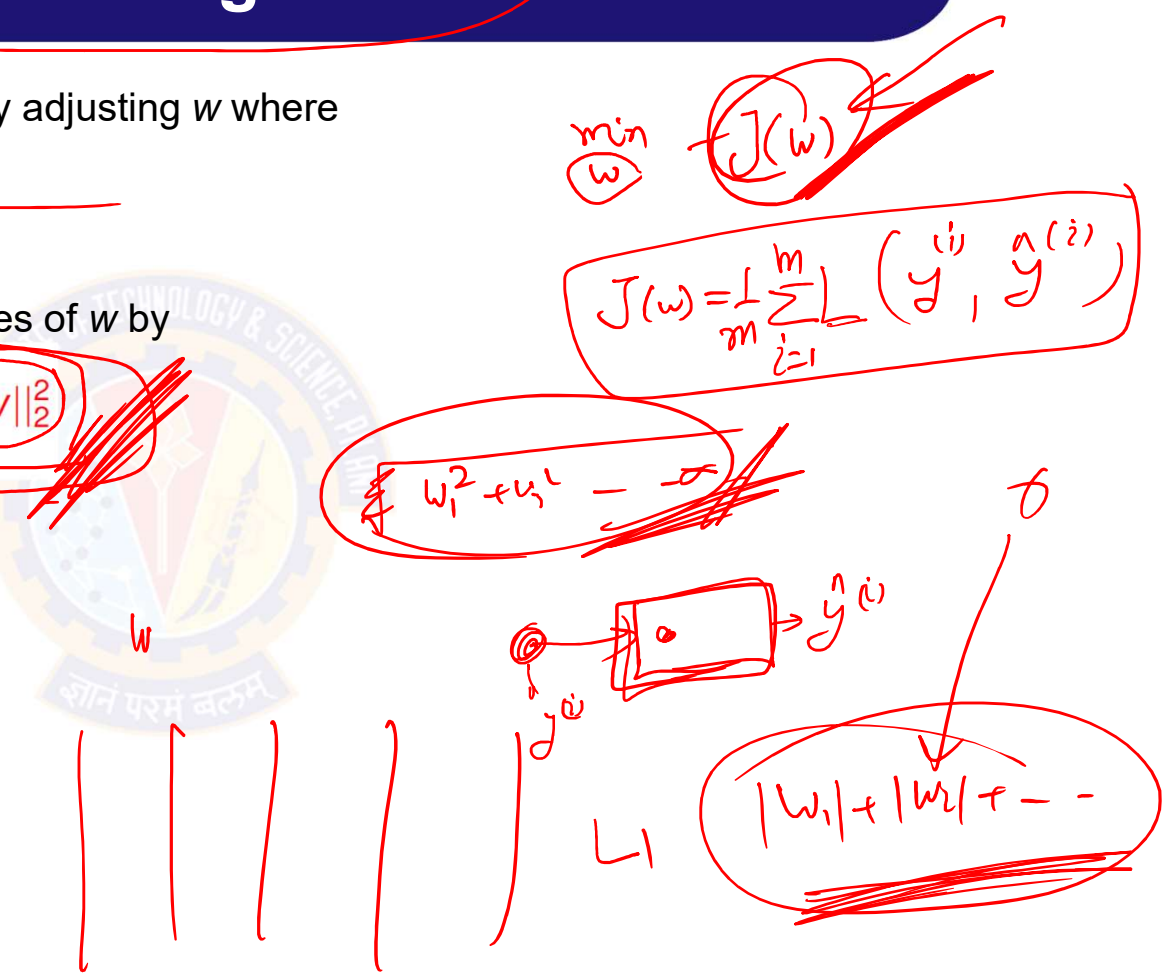
- Regularization penalizes the large values of w by

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}||w||_2^2$$

where

$$||w||_2^2 = \sum_j w_j^2 = w^T w$$

λ being regularization parameter



Regularization for NN

- As there could be L layers each with their parameters so

$$J(w^{[1]}, w^{[2]}, ..., w^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} ||w^{[l]}||_F^2$$

where frobenius norm is

$$||w^{[l]}||_F^2 = \sum_{i=1}^{n^{[l-1]}} \sum_{j=1}^{n^{[l]}} w_{ij}^2$$

- How you update the parameter earlier?

Get

$dw^{[l]}$ = (*from backpropogation*) that is $\frac{\partial J}{\partial w^{[l]}}$

then

$$w^{[l]} = w^{[l]} - \alpha . dw^{[l]}$$

Regularization for NN (contd…)

- With regularization term $dw^{[l]}$ = (*from backpropogation*) + $\frac{\lambda}{m} w^{[l]}$
- Therefore the update is modified to

$$w^{[l]} = w^{[l]} - \alpha.((\textit{from backpropogation}) + \frac{\lambda}{m} w^{[l]})$$

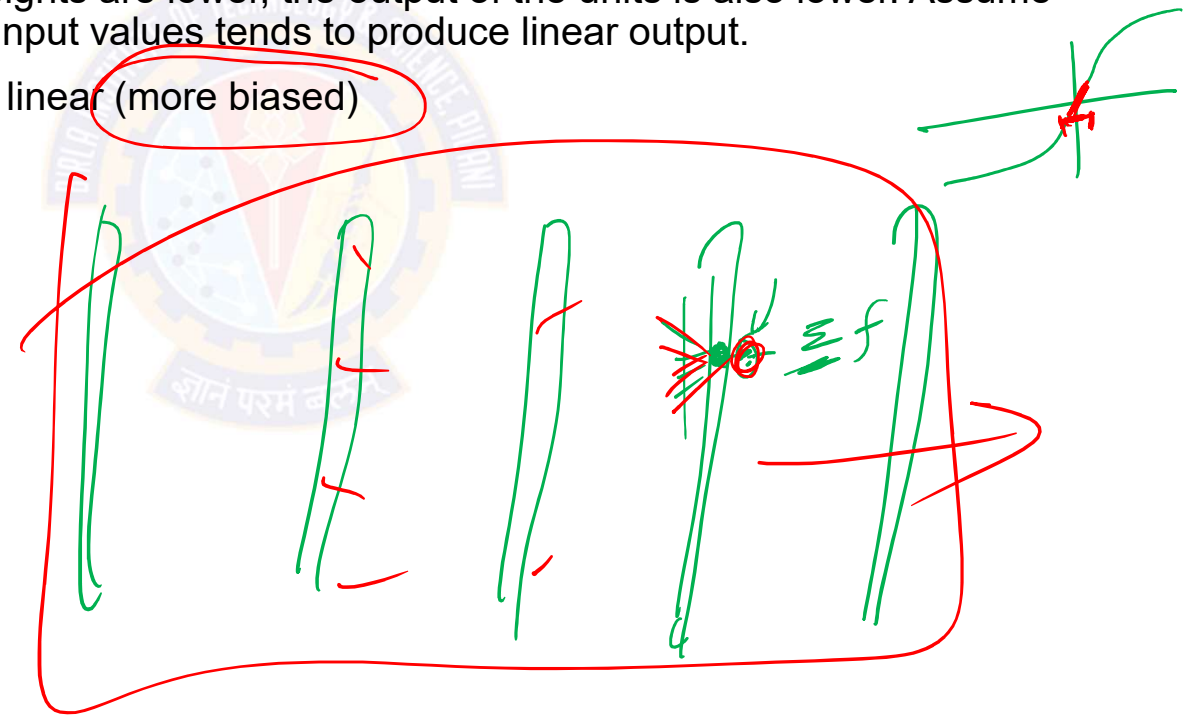
Which is

$$w^{[l]} = \left(1 - \frac{\alpha\lambda}{m}\right) w^{[l]} - \alpha.(\textit{from backpropogation})$$

- Due to the $\left(1 - \frac{\alpha\lambda}{m}\right)$ factor, this update method is also called weight decay
- Our objective here is to penalize the weight matrices being too large

Penalize the weight matrices from being too large

- With low weight, the connection get weakened so network has effectively less connections and become simpler.
- Another intuition is that, when weights are lower, the output of the units is also lower. Assume activation function be tanh small input values tends to produce linear output.
- So overall n/w tends to becomes linear (more biased)



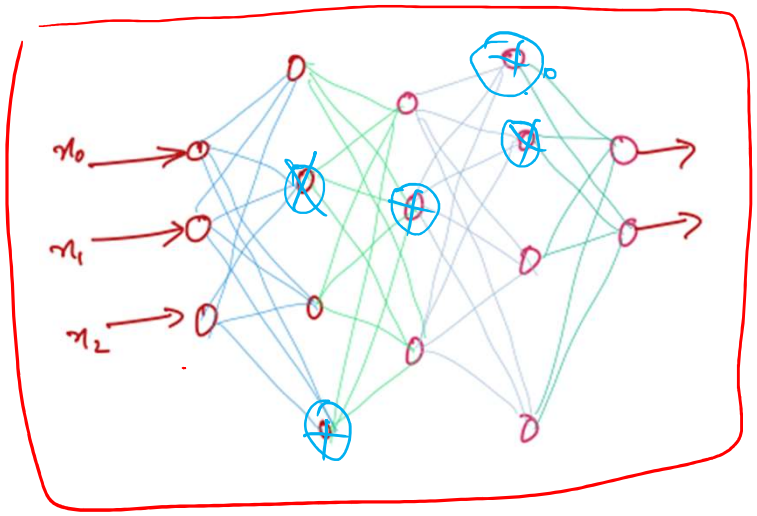
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

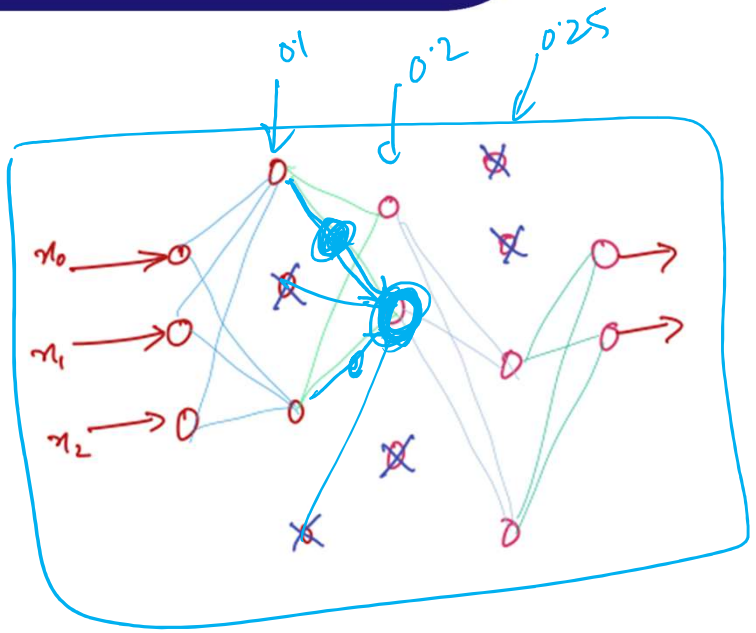
DEEP LEARNING AND ANN (Dropout)

Dr. Kamlesh Tiwari

Dropout Regularization

Randomly shutdown some of the units

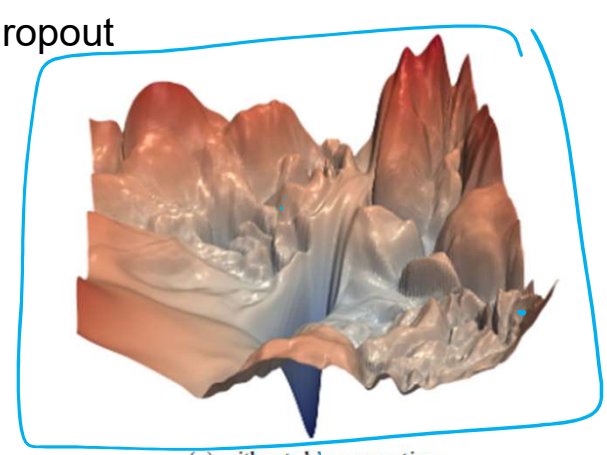




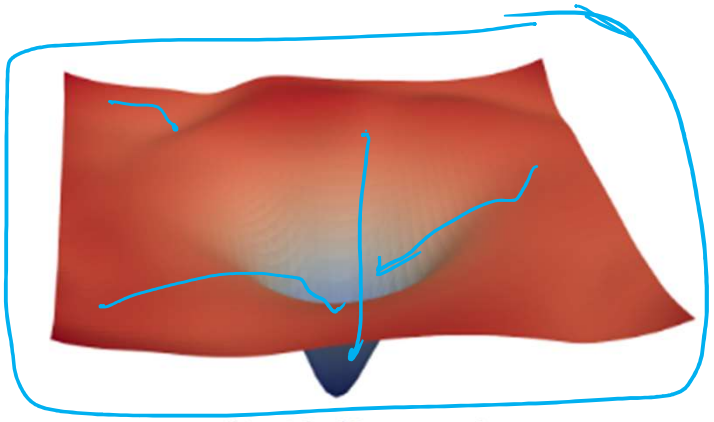
- Drop probabilities for different layers may be different
- Networks can not rely on single connection. So have to give importance to others also
- Issue is that cost function in now not well defined

Regularization

- Dropout



(a) without skip connections



(b) with skip connections

Visualizing the Loss Landscape of Neural Nets, Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein, 2018

- Early stopping

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DEEP LEARNING AND ANN
(Other methods)

Dr. Kamlesh Tiwari

Other Regularization Methods

- Increase the data by data augmentation
- Flip, rotate, scale, translate, deform ....

- Normalize training examples to speed up your training

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DEEP LEARNING AND ANN
(Weight Initialization and Training)

Dr. Kamlesh Tiwari

Weight initialization and training

- Exploding and vanishing gradient is an issue
- Set the variance of weights to be $1/n$
- For ReLu it is better to use variance as $2/n$
- For tanh use variance as $\sqrt{1/n}$ called xavier initialization
- Adam optimization, uses momentum and RMSProp simultaneously

$\beta_1, \beta_2, \epsilon = 0.00001$

iteration t

mini batch dw, db

w, b

$V_{dw}, V_{db}, S_{dw}, S_{db} = 0$

$$V_{dw} = \beta_1 V_{dw} + (1-\beta_1)\, dw$$

$$V_{db} = \beta_1 V_{db} + (1-\beta_1)\, db$$

$$S_{dw} = \beta_2 S_{db} + (1-\beta_2)\, dw^2$$

$$S_{db} = \beta_2 S_{db} + (1-\beta_2)\, db^2$$

$$S_{dw}^{corr} = \frac{S_{dw}}{(1+\beta_2^t)} \qquad V_{dw}^{corr} = \frac{V_{dw}}{(1+\beta_1^t)}$$

$$S_{db}^{corr} = \frac{S_{db}}{(1+\beta_2^t)} \qquad V_{dw}^{corr} = \frac{V_{db}}{(1+\beta_1^t)}$$

$$w = w - \alpha \frac{V_{dw}^{corr}}{\sqrt{S_{dw}^{corr}} + \epsilon} \qquad b = b - \alpha \frac{V_{db}^{corr}}{\sqrt{S_{db}^{corr}} + \epsilon}$$

$w = w - \alpha V_{dw}$

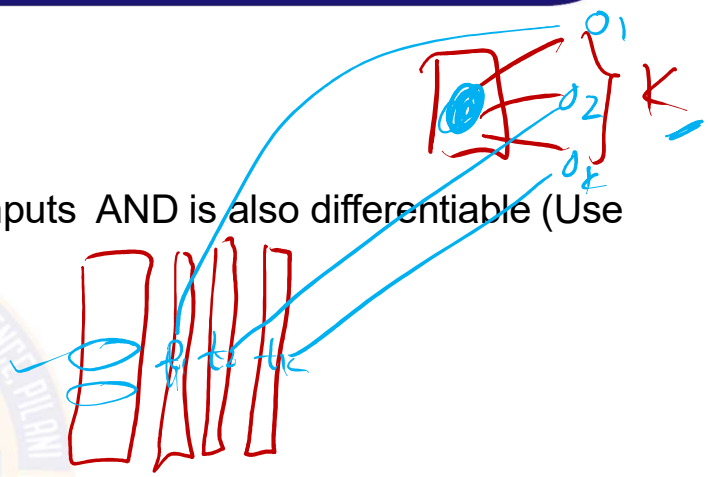
$b = b - \alpha V_{db}$

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Backpropogation

Kamlesh Tiwari

Multilayer Networks and Backpropagation



- Single perceptron can only express linear decision surface
- We need units whose output is a nonlinear function of its inputs AND is also differentiable (Use Neuron not ~~Perceptron~~)

$$o(\vec{x}) = \sigma(\vec{w}.\vec{x})$$

where $\sigma(y) = \frac{1}{1+e^{-y}}$

- Backpropagation algorithm learns the weights for a fixed set of units and interconnections
- It employs gradient descent to minimize the error between the network output values and the target values for these outputs
- Let Error function is redefined as

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

Backpropagation (for 2 layers)

Algorithm 12: Backpropagation(D, η, n_{in}, n_{out}, n_{hidden})

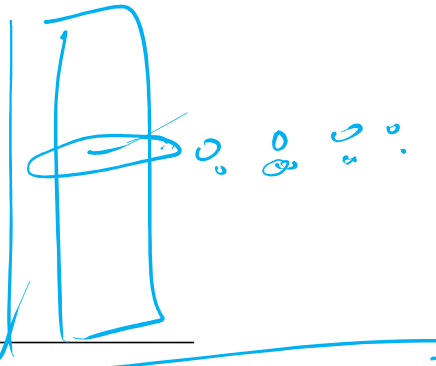
1 Create the feedforward network with n_{in} , n_{out} , n_{hidden} layers
2 Randomly initialize weights to small values $\in [-0.05, +0.05]$
3 **repeat**
4 **for** *each* $\langle \vec{x}, \vec{t} \rangle \in D$ **do**
5 $o_u =$ get output from network $\forall$ unit u
6 $\delta_k = o_k(1 - o_k)(t_k - o_k)$ for all output unit k
7 $\delta_h = o_h(1 - o_h) \sum_{k \in outputs}(w_{kh}\delta_k)$ for all hidden unit h
8 $w_{ji} = w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$
9 **until** *converge*;

- Recall error function is $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$

Backpropagation (for 2 layers)

Algorithm 13: Backpropagation(D, η, n_{in}, n_{out}, n_{hidden})

Create the feedforward network with n_{in} , n_{out} , n_{hidden} layers
Randomly initialize weights to small values $\in [-0.05, +0.05]$
repeat
  for *each* $\langle \vec{x}, \vec{t} \rangle \in D$ **do**
    $o_u =$ get output from network $\forall$ unit u
    $\delta_k = o_k(1 - o_k)(t_k - o_k)$ for all output unit k
    $\delta_h = o_h(1 - o_h) \sum_{k \in outputs}(w_{kh}\delta_k)$ for all hidden unit h
    $w_{ji} = w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$
until *converge*;



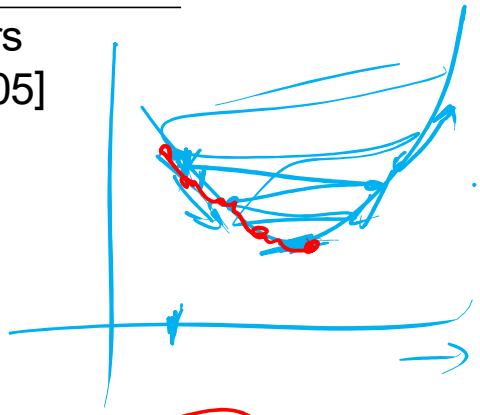
- Recall error function is $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs}(t_{kd} - o_{kd})^2$
- For a single training example $E_d(\vec{w}) = \frac{1}{2} \sum_{k \in outputs}(t_k - o_k)^2$

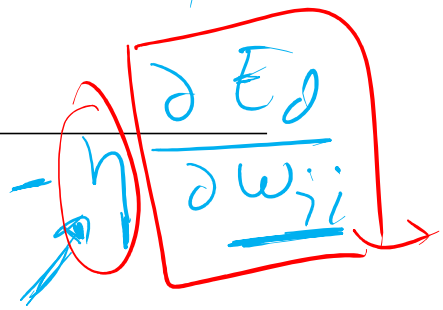
$$E(\vec{w}) = \sum_{d \in D} E_d(\vec{w})$$

Backpropagation (for 2 layers)

Algorithm 14: Backpropagation(D,η,n_{in},n_{out},n_{hidden})

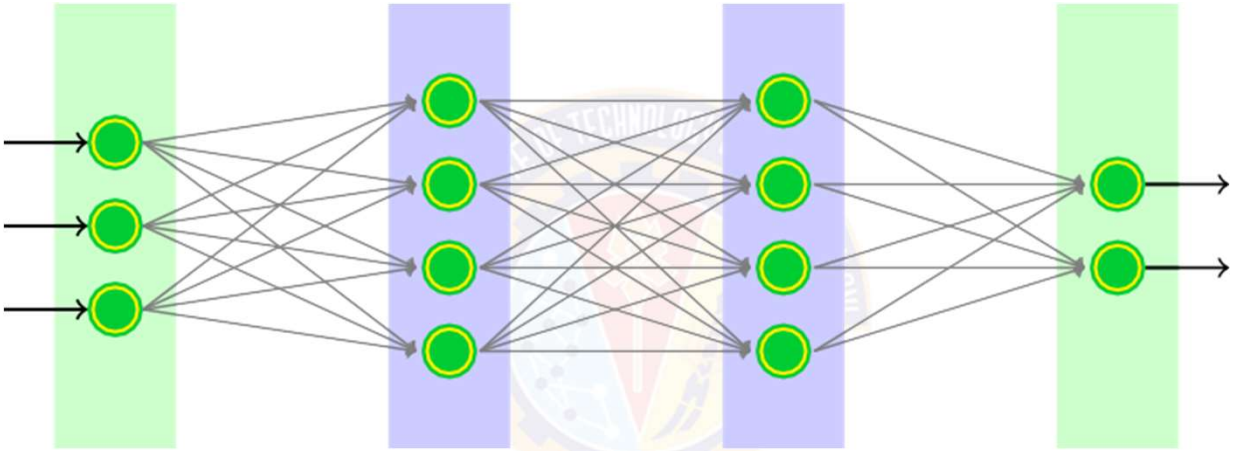
Create the feedforward network with n_{in} , n_{out} , n_{hidden} layers
Randomly initialize weights to small values $\in[-0.05, +0.05]$
repeat
  for *each* $<\vec{x},\vec{t}> \in D$ **do**
    $o_u =$ get output from network $\forall$ unit u
    $\delta_k = o_k(1 - o_k)(t_k - o_k)$ for all output unit k
    $\delta_h = o_h(1 - o_h)\sum_{k \in outputs}(w_{kh}\delta_k)$ for all hidden unit h
    $w_{ji} = w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$
until *converge*;





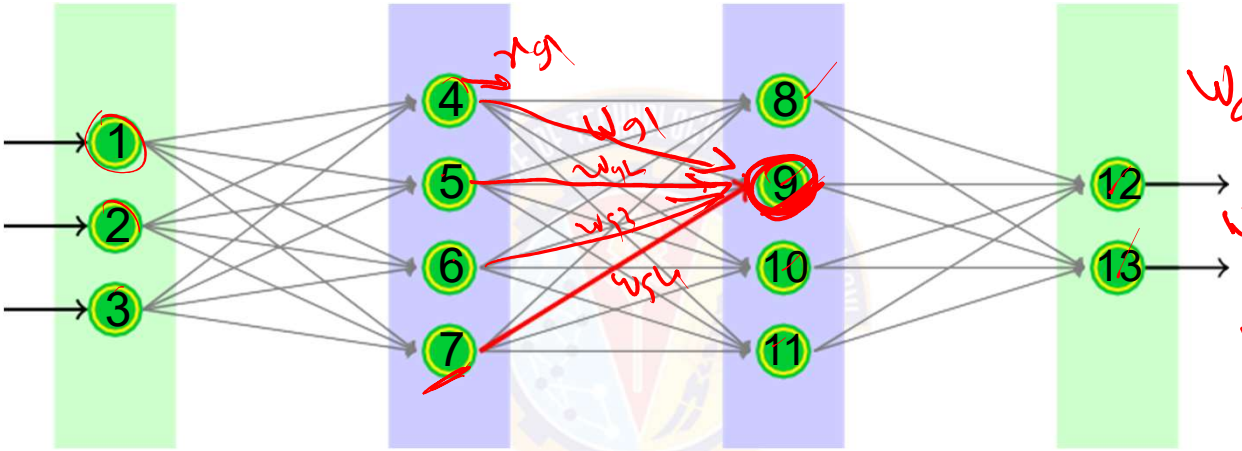
- Recall error function is $E(\vec{w}) = \frac{1}{2}\sum_{d \in D}\sum_{k \in outputs}(t_{kd} - o_{kd})^2$
- For a single training example, $E_d(\vec{w}) = \frac{1}{2}\sum_{k \in outputs}(t_k - o_k)^2$
- Weight w_{ji} is updated by adding $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$

Conventions Over The Network



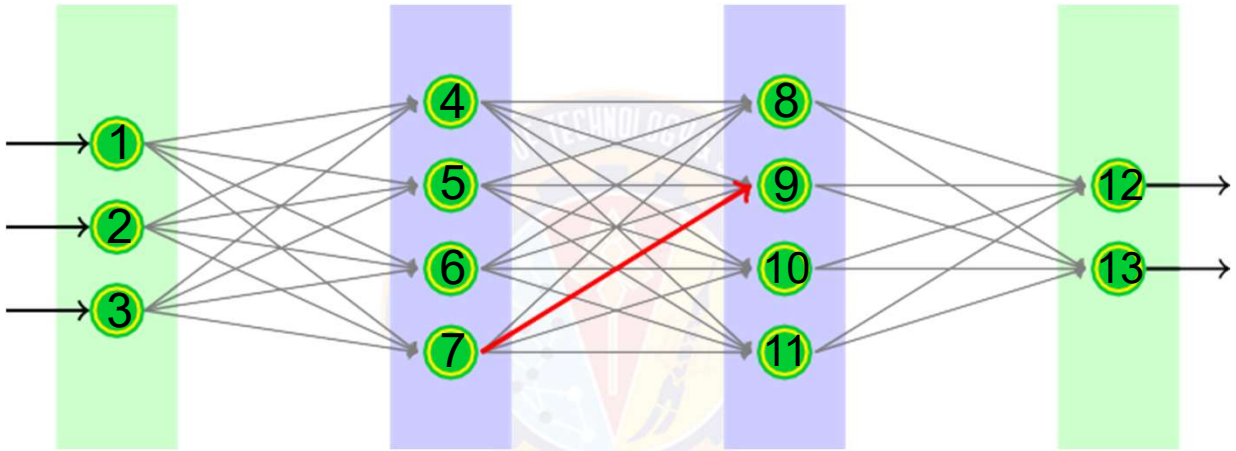
x_{ji} i th input to unit j (x_{94} is highlighted)

Conventions Over The Network



x_{ji} *i* th input to unit *j* (x_{94} is highlighted)

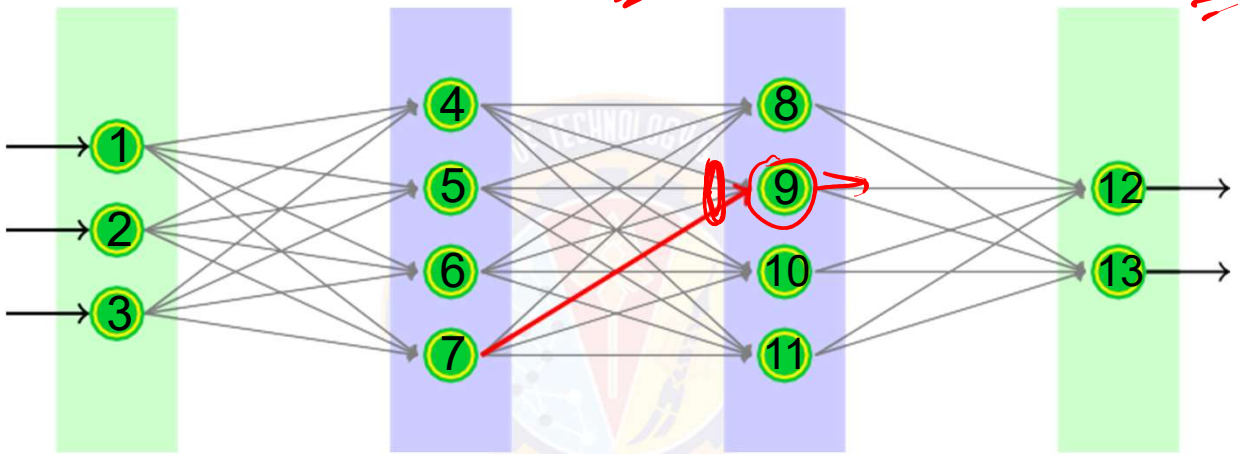
Conventions Over The Network



x_{ji} *i* th input to unit *j* (x_{94} is highlighted)

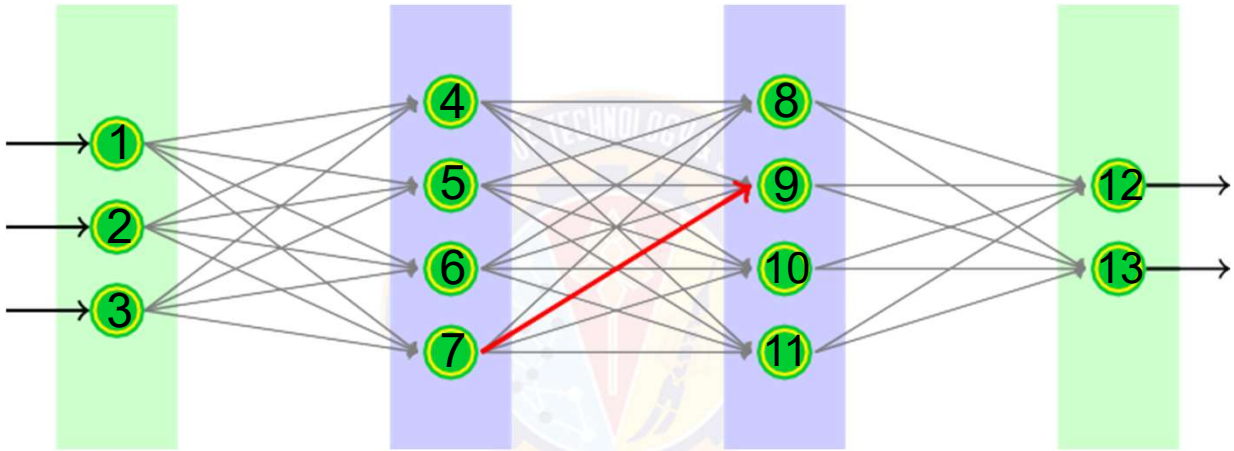
w_{ji} weight associated with *i* th input to unit*j*

Conventions Over The Network



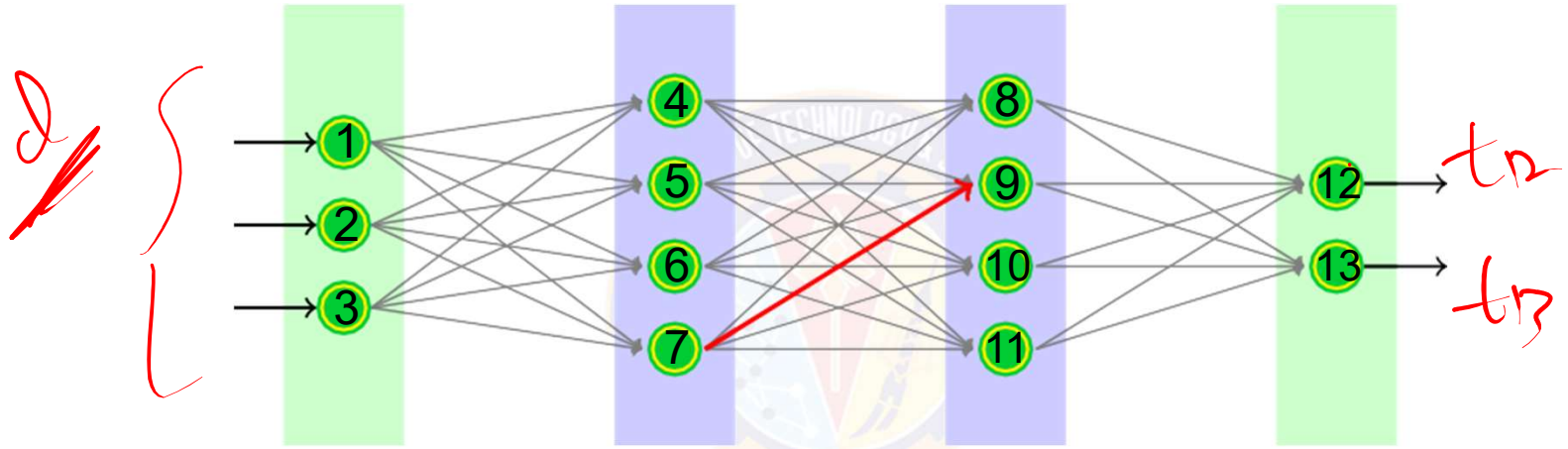
x_{ji}	i th input to unit j (x_{94} is highlighted)
w_{ji}	weight associated with i th input to unit$j$
net_j	be $\sum_i w_{ji} x_{ji}$ the weighted sum of input for unit j

Conventions Over The Network



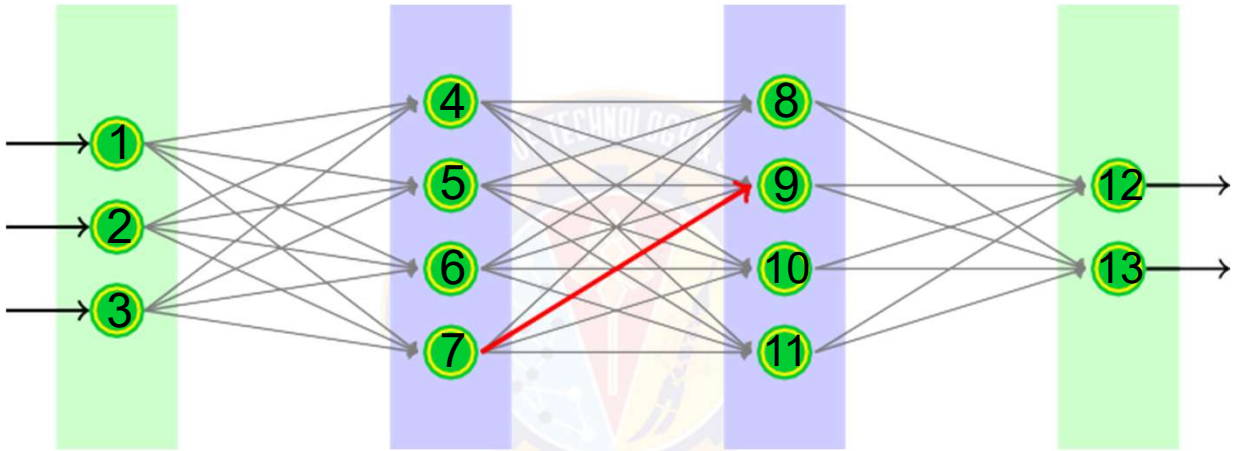
x_{ji}	i th input to unit j (x_{94} is highlighted)
w_{ji}	weight associated with i th input to unit$j$
net_j	be $\sum_i w_{ji} x_{ji}$ the weighted sum of input for unit j
o_j	output computed by unit j . Let it be $\sigma(net_j)$

Conventions Over The Network



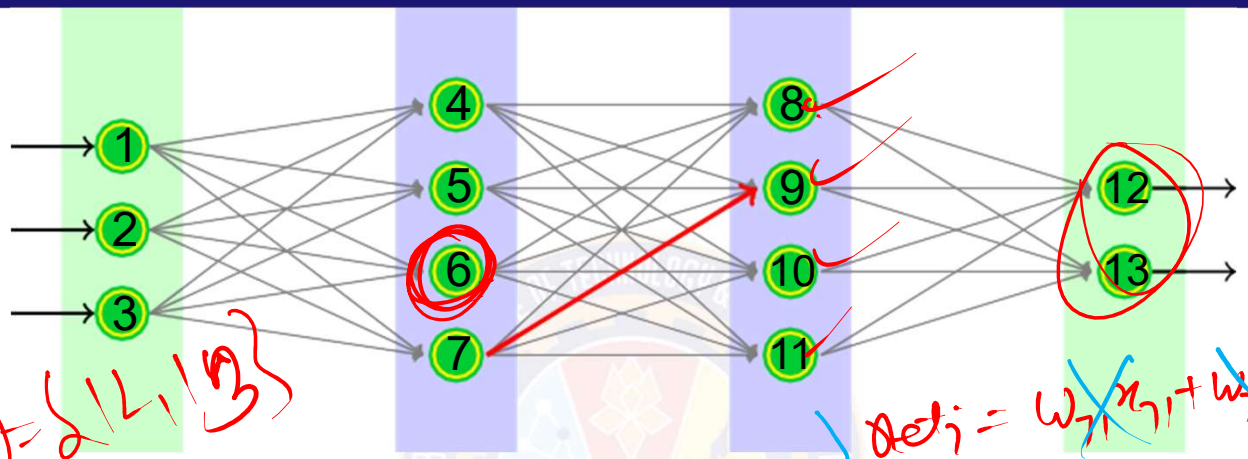
x_{ji}	i th input to unit j (x_{94} is highlighted)
w_{ji}	weight associated with i th input to unit j
net_j	be $\sum_i w_{ji} x_{ji}$ the weighted sum of input for unit j
o_j	output computed by unit j . Let it be $\sigma(net_j)$
t_j	target output for unit j

Conventions Over The Network



x_{ji}	i th input to unit j (x_{94} is highlighted)
w_{ji}	weight associated with i th input to unit j
net_j	be $\sum_i w_{ji} x_{ji}$ the weighted sum of input for unit j
o_j	output computed by unit j . Let it be $\sigma(net_j)$
t_j	target output for unit j
outputs	set of units in final layer ({12, 13} in our case)

Conventions Over The Network



x_{ji}	i th input to unit j (x_{94} is highlighted)
w_{ji}	weight associated with i th input to unit j
net_j	be $\sum_i w_{ji} x_{ji}$ the weighted sum of input for unit j
o_j	output computed by unit j . Let it be $\sigma(net_j)$
t_j	target output for unit j
outputs	set of units in final layer ($\{12, 13\}$ in our case)
Downstream(j)	units whose immediate input is the output of unit j

We are interested in $\frac{\partial E_d}{\partial w_{ji}}$ it is $\frac{\partial E_d}{\partial net_j} \times \frac{\partial net_j}{\partial w_{ji}}$ and therefore, $\frac{\partial E_d}{\partial net_j} \times x_{ji}$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$o_j = \sigma(net_j)$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2$$

$$\frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2$$

$$\frac{1}{2} \frac{\partial}{\partial o_j} (t_j - o_j)^2$$

$$\frac{1}{2} \times 2 (t_j - o_j) \frac{\partial}{\partial o_j} (t_j - o_j)$$

$$(-1)(t_j - o_j)$$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

$$\frac{\partial}{\partial net_j} o_j = \frac{\partial}{\partial net_j} \sigma(net_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\frac{d}{dx}\sigma(x) = \frac{d}{dx}\frac{1}{1+e^{-x}}$$

$(1+e^{-x})^{-1}$

$(-1)(1+e^{-x})^{-2} \frac{\partial}{\partial x}(1+e^{-x})$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\frac{d}{dx}\sigma(x) = \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x})$$

$$\left(\frac{0+d}{dx}e^{-x}\right)$$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \end{aligned}$$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} \end{aligned}$$

$$\left(\frac{1}{1+e^{-x}} \times \frac{e^{-x}+1}{1+e^{-x}} = \frac{1}{1+e^{-x}}\right)$$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} = \sigma(x)(1-\sigma(x)) \end{aligned}$$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} = \sigma(x)(1-\sigma(x)) \end{aligned}$$

As a result $\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = \sigma(net_j)(1 - \sigma(net_j))$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} = \sigma(x)(1-\sigma(x)) \end{aligned}$$

As a result $\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = \sigma(net_j)(1 - \sigma(net_j)) = o_j(1 - o_j)$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} = \sigma(x)(1-\sigma(x)) \end{aligned}$$

As a result $\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = \sigma(net_j)(1 - \sigma(net_j)) = o_j(1 - o_j)$

$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j)$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} = \sigma(x)(1-\sigma(x)) \end{aligned}$$

As a result $\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = \sigma(net_j)(1 - \sigma(net_j)) = o_j(1 - o_j)$

$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)o_j(1 - o_j$

Term $(t_j - o_j)o_j(1 - o_j)$ is treated as δ_j

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} = \sigma(x)(1-\sigma(x)) \end{aligned}$$

As a result $\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = \sigma(net_j)(1 - \sigma(net_j)) = o_j(1 - o_j)$

$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j$

Term $(t_j - o_j) o_j (1 - o_j)$ is treated as δ_j

Therefore, $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} \times x_{ji}$

Value of $\frac{\partial Ed}{\partial net_j}$ for output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = -(t_j - o_j)$$

Note that $o_j = \sigma(net_j)$ therefore $\frac{\partial o_j}{\partial net_j}$ is derivative of sigmoid

$$\begin{aligned} \frac{d}{dx}\sigma(x) &= \frac{d}{dx}\frac{1}{1+e^{-x}} = (-1)(1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x}) \\ &= (-1)(1+e^{-x})^{-2}(0-e^{-x}) \\ &= \frac{1}{1+e^{-x}} \times \frac{e^{-x}+1-1}{1+e^{-x}} = \sigma(x)(1-\sigma(x)) \end{aligned}$$

As a result $\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = \sigma(net_j)(1 - \sigma(net_j)) = o_j(1 - o_j)$

$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j) o_j (1 - o_j)$

Term $(t_j - o_j) o_j (1 - o_j)$ is treated as δ_j

$\frac{\partial E_d}{\partial net_j} = -\delta_j$

Therefore, $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} \times x_{ji} = \boxed{\eta (t_j - o_j) o_j (1 - o_j) x_{ji}}$

Value of $\frac{\partial E_d}{\partial net_j}$ for hidden units

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \times \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \times \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \times \frac{\partial net_k}{\partial o_j} \times \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \times w_{kj} \times o_j(1 - o_j)$$

δ_j being $-\frac{\partial E_d}{\partial net_j} = o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k \times w_{kj}$

Therefore, $\Delta w_{ji} = \eta \delta_j x_{ji} = \boxed{\eta(o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k \times w_{kj}) x_{ji}}$

Backpropagation (for 2 layers)

Algorithm 14: Backpropagation(D,η,n_{in},n_{out},n_{hidden})

1 Create the feedforward network with n_{in} , n_{out} , n_{hidden} layers
2 Randomly initialize weights to small values $\in[-0.05, +0.05]$
3 **repeat**
4 **for** *each* $\langle\vec{x},\vec{t}\rangle \in D$ **do**
5 o_u = get output from network $\forall$ unit u
6 $\delta_k = o_k(1 - o_k)(t_k - o_k)$ for all output unit k
7 $\delta_h = o_h(1 - o_h)\sum_{k\in outputs}(w_{kh}\delta_k)$ for all hidden unit h
8 $w_{ji} = w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta\delta_j x_{ji}$
9 **until** *converge*;

Backpropagation

- **Adding Momentum:** weight update during n^{th} iteration depend partially on the update that occurred during the $(n-1)^{th}$ iteration

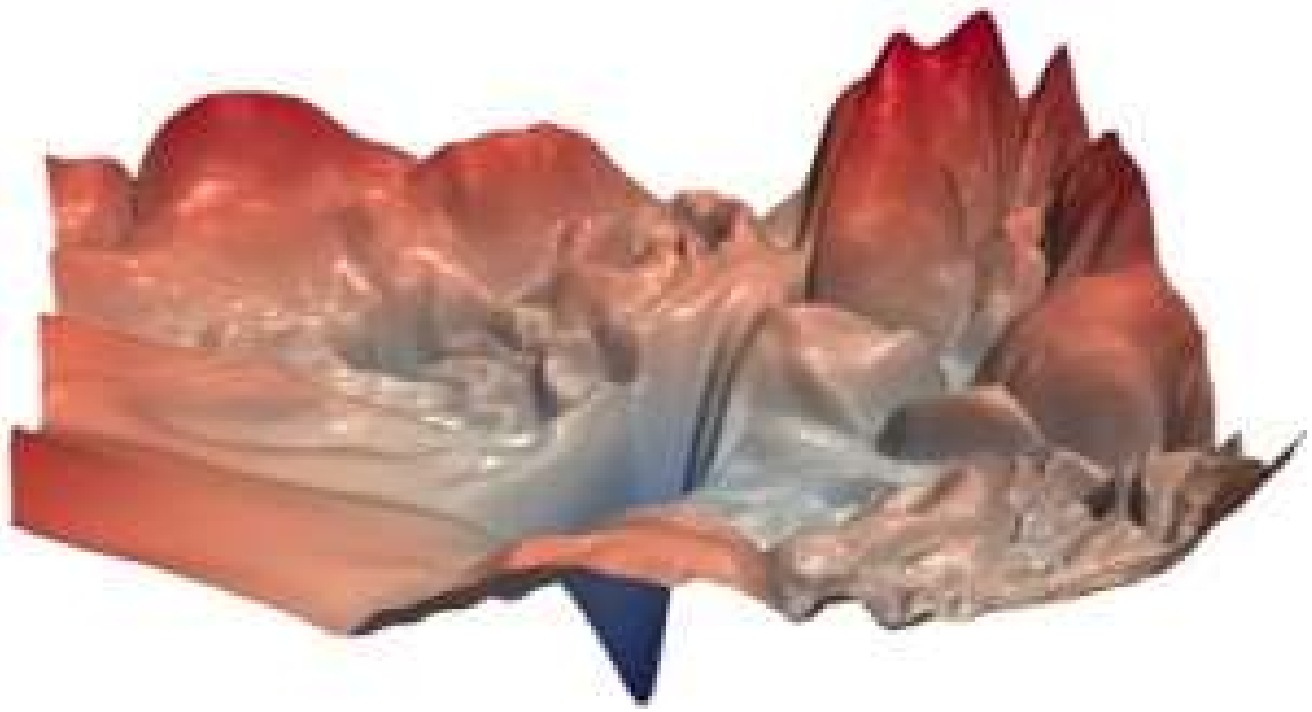
$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$$

- **Learning in arbitrary acyclic network:** for feed-forward networks of arbitrary depth, δ_r value for a unit in hidden layer is determined as

$$\delta_r = o_r(1 - o_r) \sum_{s \in Downstream(r)} w_{sr} \times \delta_s$$

Backpropagation

- Result of Backpropagation over multilayer networks is only guaranteed to converge toward some local minimum and not necessarily to the global minimum error



Backpropagation

- Result of Backpropagation over multilayer networks is only guaranteed to converge toward some local minimum and not necessarily to the global minimum error
- No methods are known to predict with certainty when local minima will cause difficulties
- Suggested to use momentum, true gradient descent or multiple networks (initialized with different random weights)
- Any boolean function can precisely be represented by some network having only **two** layers of units (Cybenko 1989; Hornik et al. 1989)

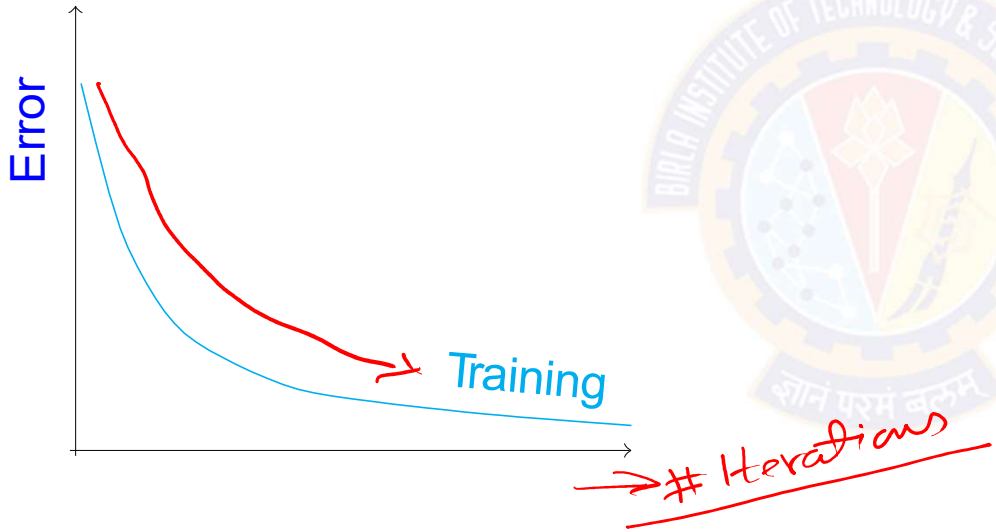
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

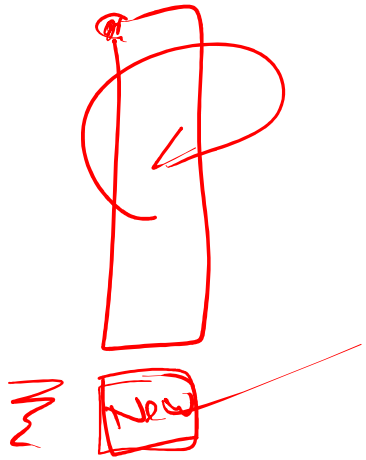
Generalization Overfitting and Stopping Criteria

Kamlesh Tiwari

Generalization, Overfitting, and Stopping Criterion

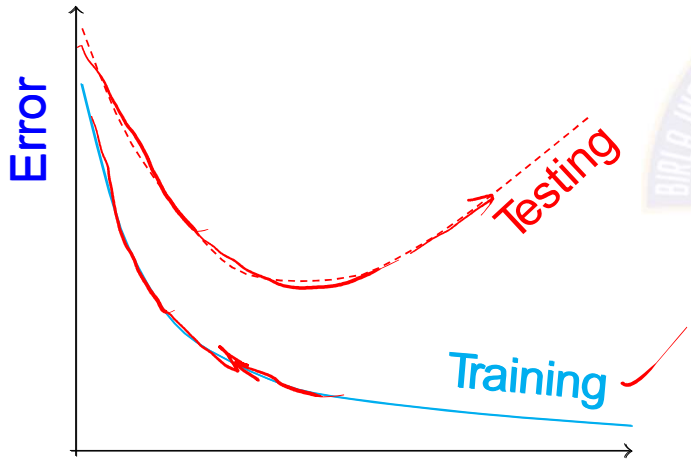
Continue training until the error on the training examples falls below some predetermined threshold could be a poor strategy





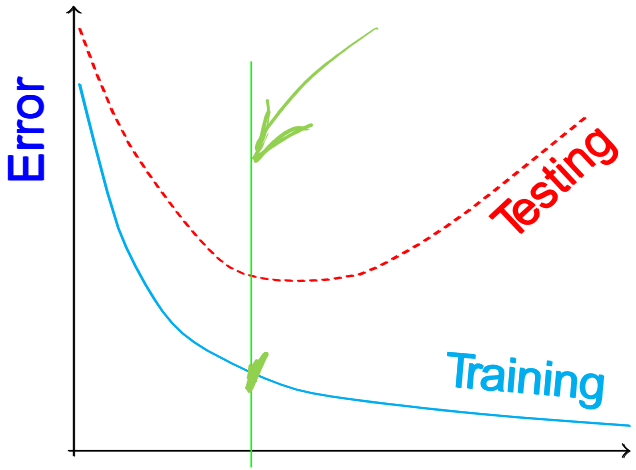
Generalization, Overfitting, and Stopping Criterion

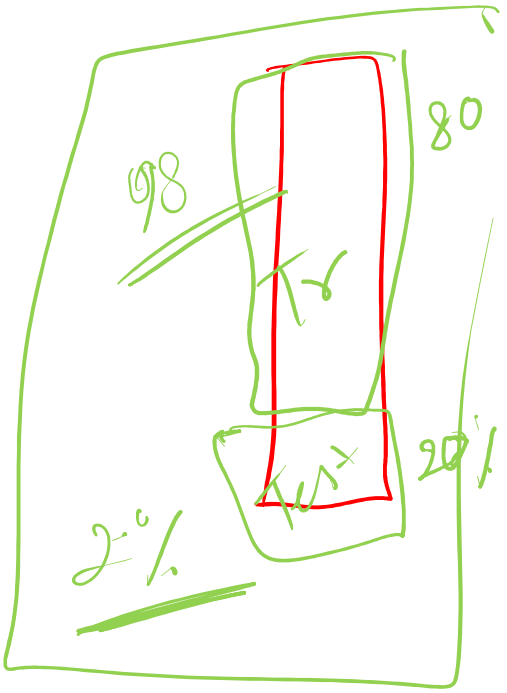
Continue training until the error on the training examples falls below some predetermined threshold could be a poor strategy



Generalization, Overfitting, and Stopping Criterion

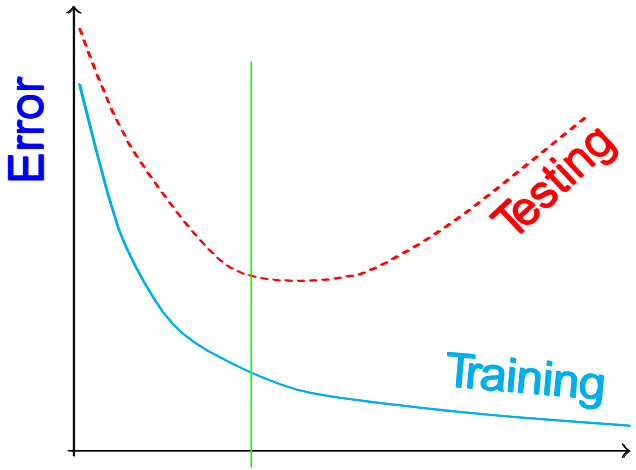
Continue training until the error on the training examples falls below some predetermined threshold could be a poor strategy





Generalization, Overfitting, and Stopping Criterion

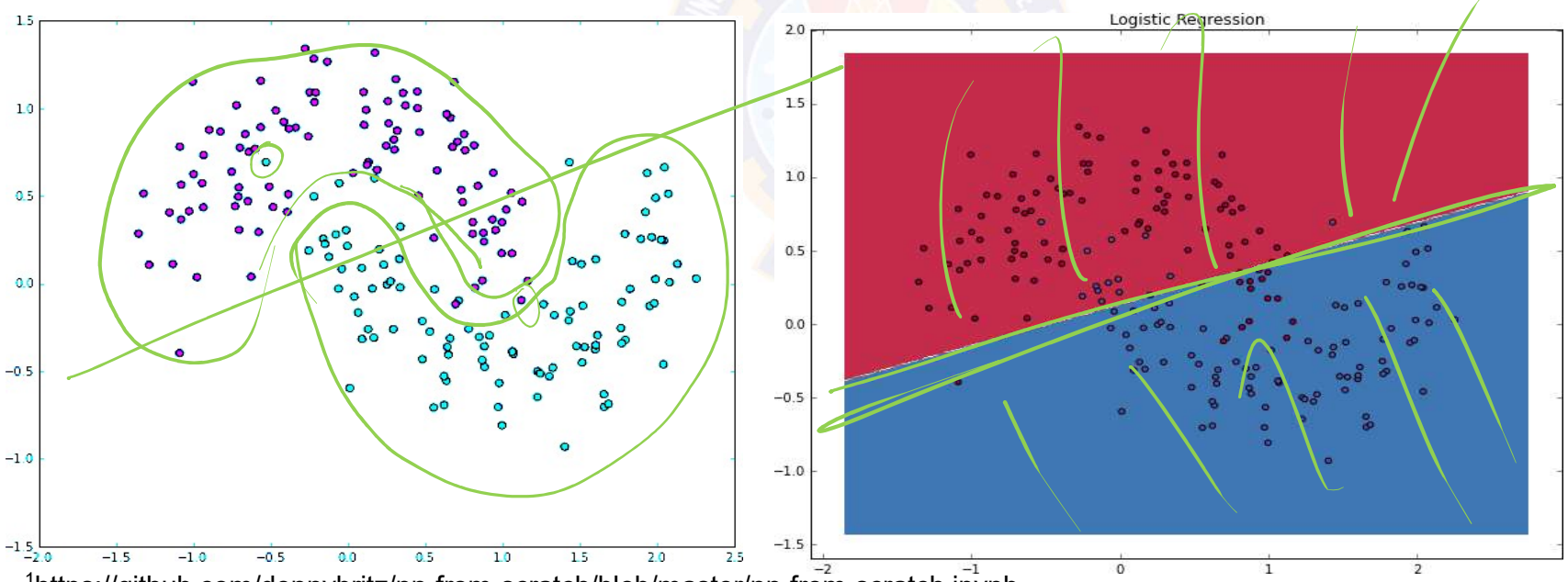
Continue training until the error on the training examples falls below some predetermined threshold could be a poor strategy



- Weight decay or use of validation set (*k* -fold ?) is suggested
- Input or output encoding can be used

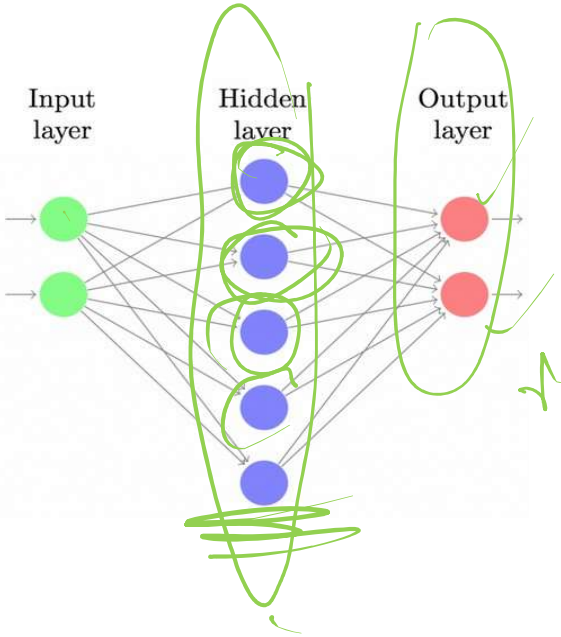
Coding Example [1]

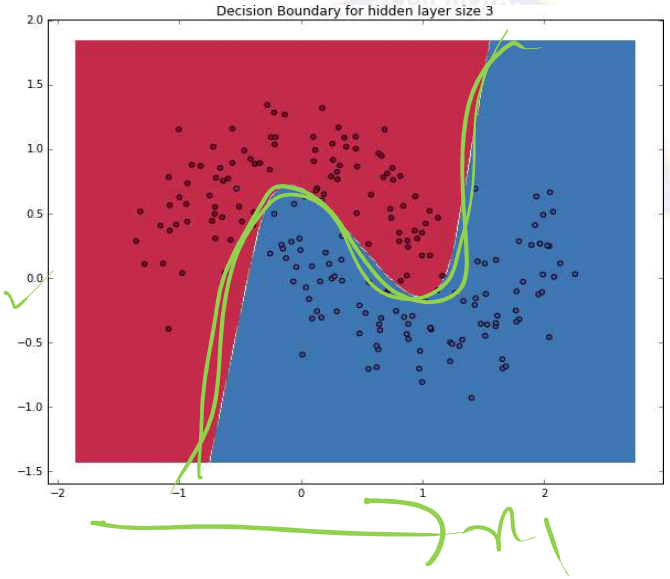
Consider two class data
Logistic Regression
Neural Network
Decision Boundary
Bigger hidden layer

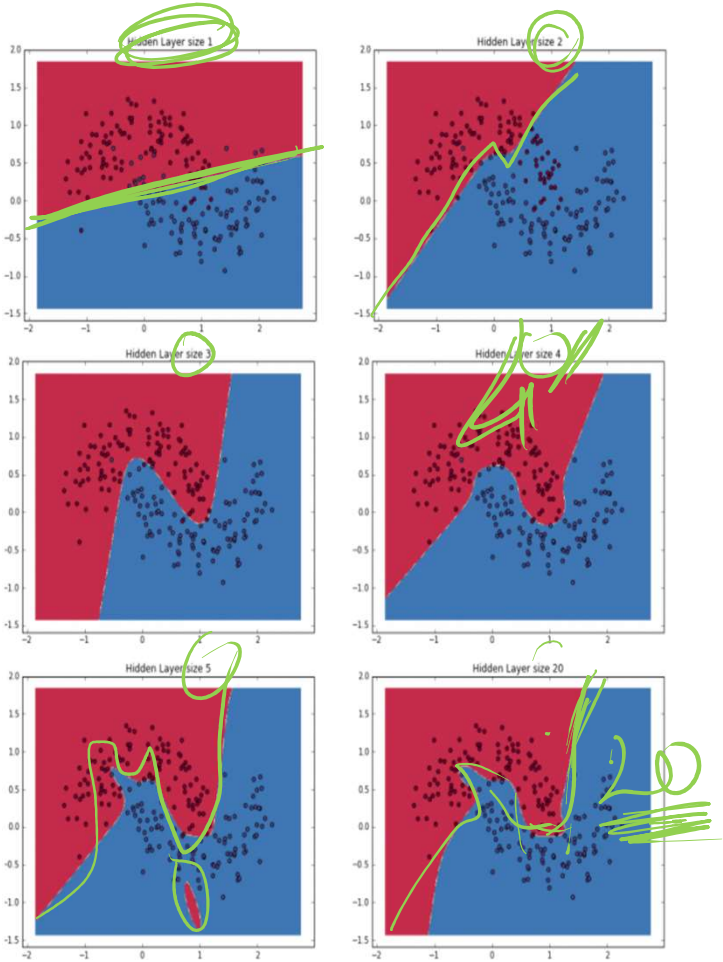


[1]https://github.com/dennybritz/nn-from-scratch/blob/master/nn-from-scratch.ipynb

Coding Example [1]







[1] https://github.com/dennybritz/nn-from-scratch/blob/master/nn-from-scratch.ipynb

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

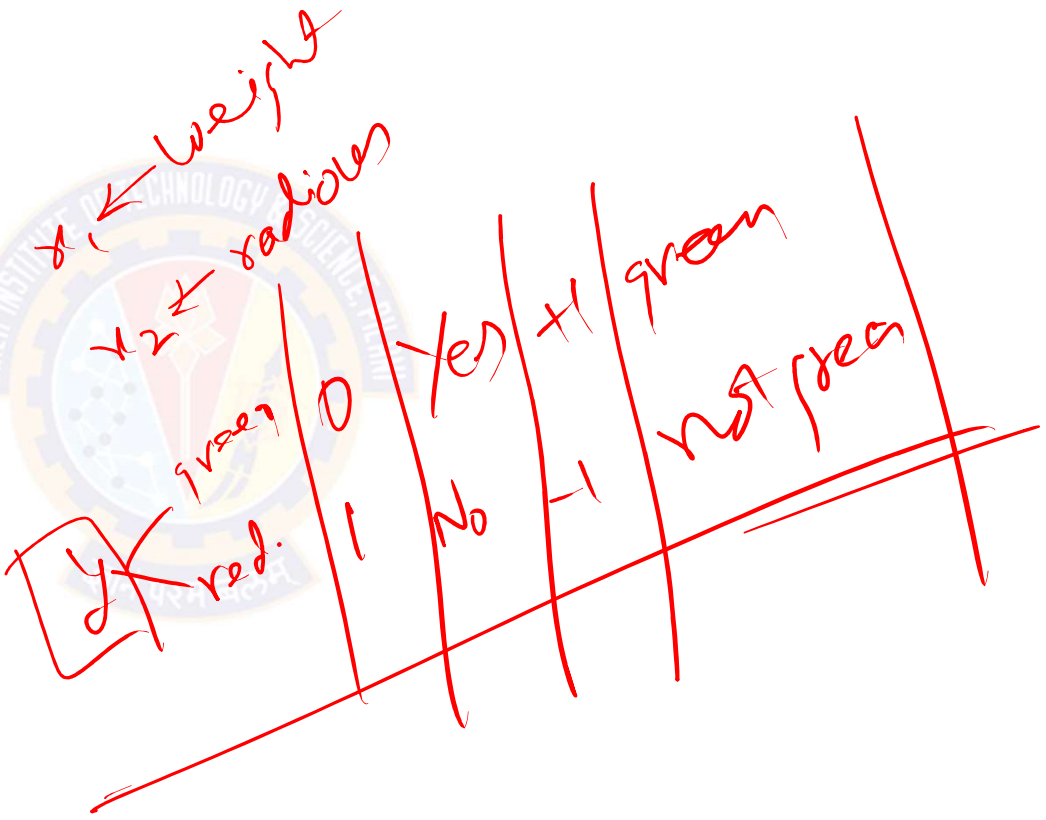
Background of NN

Dr. Kamlesh Tiwari

Linear Classification

Consider Following data

x_1	x_2	y
1	9	Green
10	9	Green
4	7	Green
4	5	Red
5	3	Red
8	9	Green
4	2	Red
2	5	Red
7	1	Red
2	10	Green
8	5	Green
1	2	Red
8	2	Red

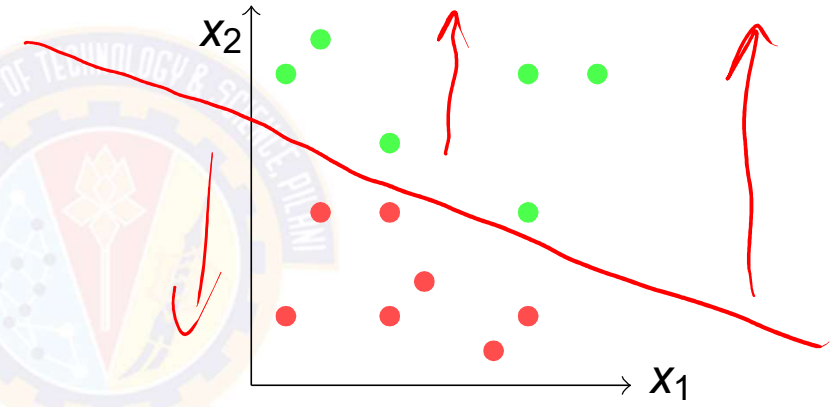


Linear Classification

Consider Following data

x_1	x_2	y
1	9	Green
10	9	Green
4	7	Green
4	5	Red
5	3	Red
8	9	Green
4	2	Red
2	5	Red
7	1	Red
2	10	Green
8	5	Green
1	2	Red
8	2	Red

Data is in 2D, so let us visualize

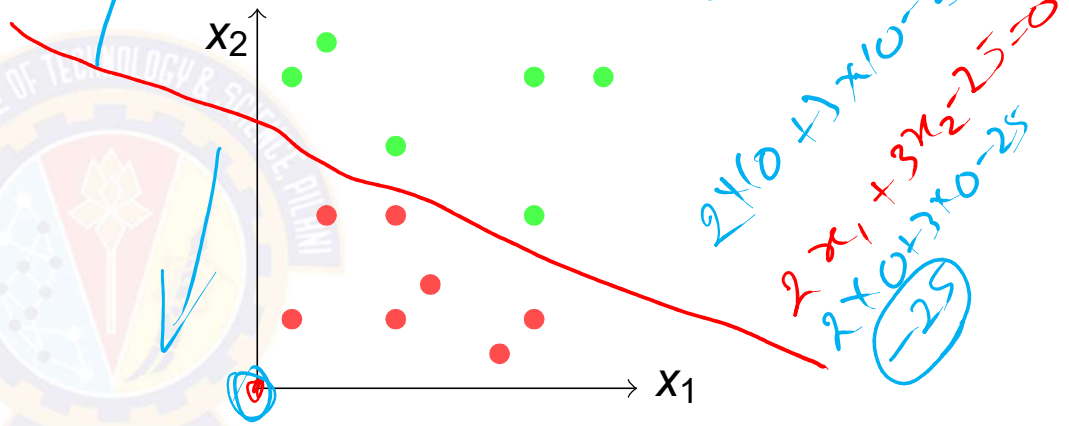


Linear Classification

Consider Following data

x_1	x_2	y
1	9	Green
10	9	Green
4	7	Green
4	5	Red
5	3	Red
8	9	Green
4	2	Red
2	5	Red
7	1	Red
2	10	Green
8	5	Green
1	2	Red
8	2	Red

Data is in 2D, so let us visualize



- Data looks linearly separable
- What is the decision boundary?

Many Possibilities, such as

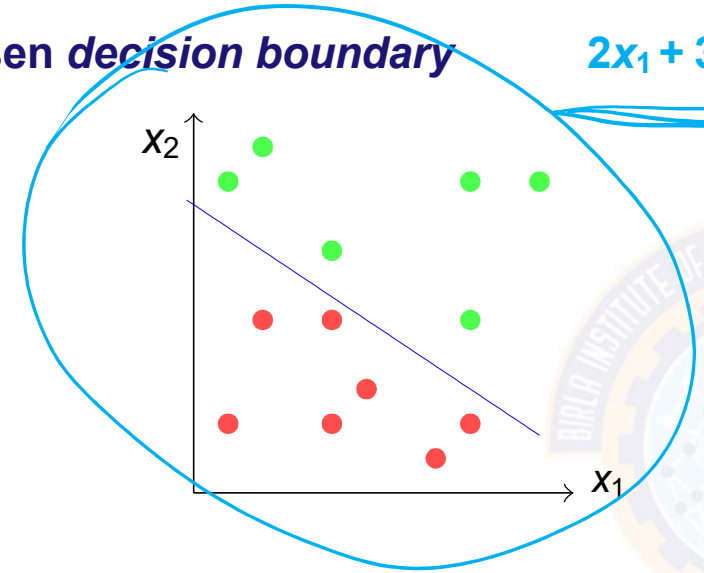
if $(2x_1 + 3x_2 - 25 > 0)$ it is green

otherwise red

What about this arrangement?

With chosen *decision boundary*

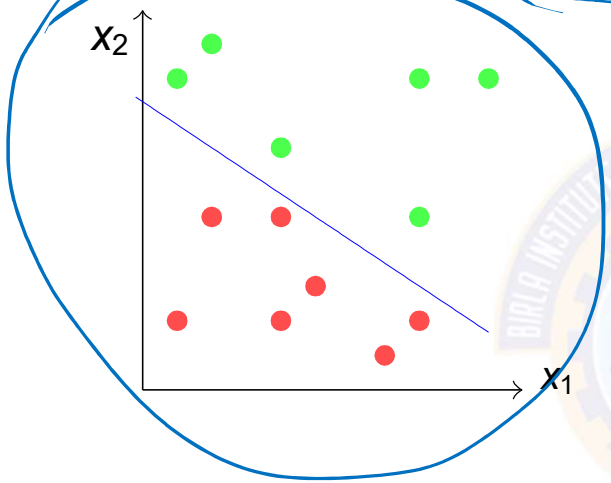
$2x_1 + 3x_2 - 25 = 0$

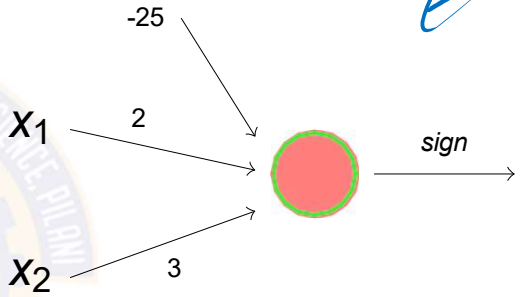


What about this arrangement?

With chosen *decision boundary*

$2x_1 + 3x_2 - 25 = 0$

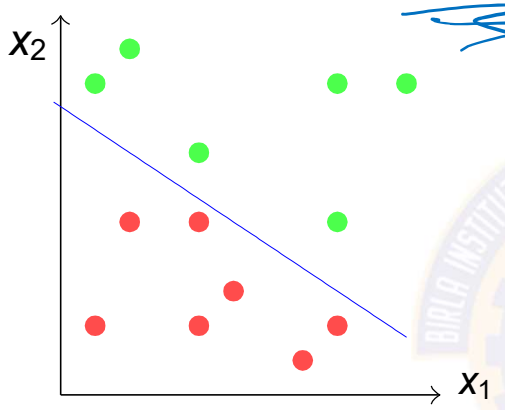


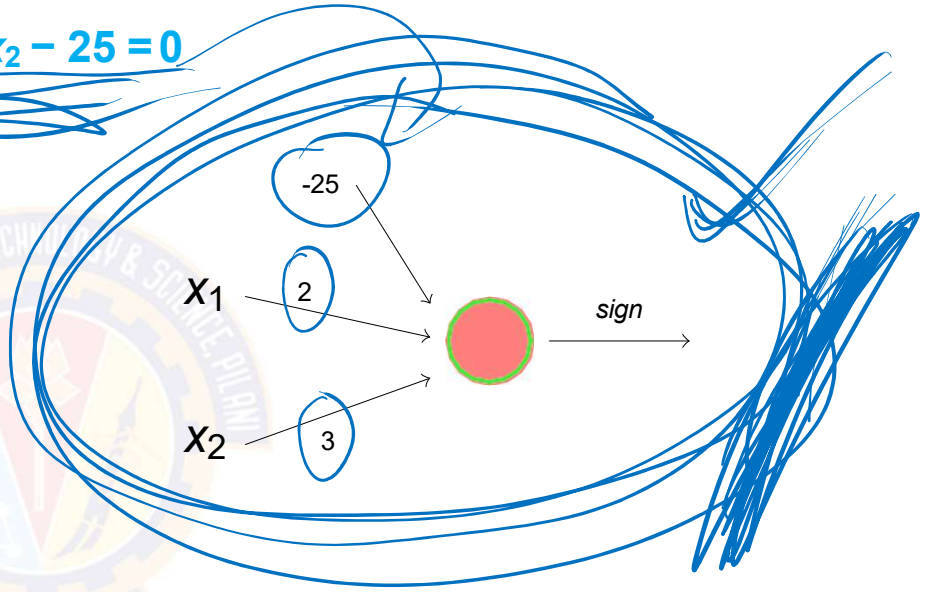


What about this arrangement?

With chosen *decision boundary*

$2x_1 + 3x_2 - 25 = 0$





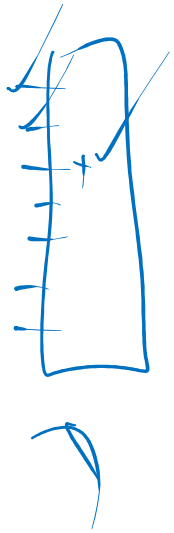
- This illustration is called as **perceptron**
- Provides a graphical way to represent the linear boundary
- Values 3, 2, -25 are its parameters or weights

Given a data

"How to find appropriate parameters?" is an important issue

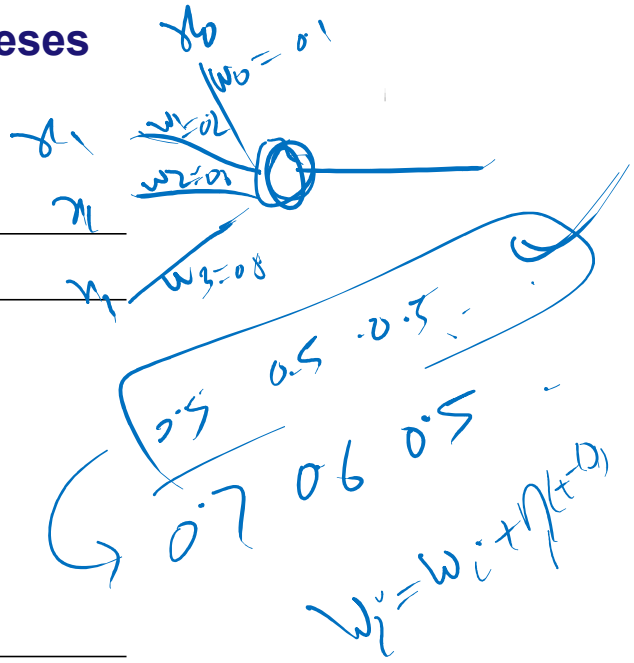
Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses



Algorithm 1: Perceptron training rule

1 Begin with random weights w
2 **repeat**
3 **for** *each misclassified example* **do**
4 $w_i = w_i + \eta(t - o)x_i$
5 **until** *all training examples are correctly classified*;
6 **return** w



Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 2: Perceptron training rule

```
1 Begin with random weights w
2 repeat
3     for each misclassified example do
4         w_i = w_i + η(t − o)x_i
5 until all training examples are correctly classified;
6 return w
```

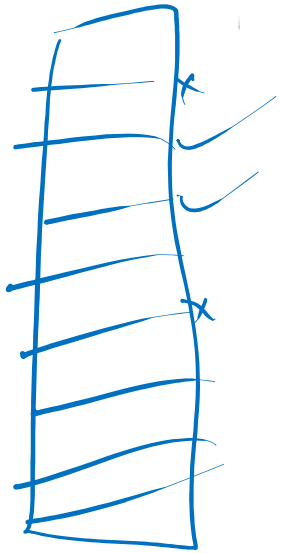
- **Why would this strategy converge?**

Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 2: Perceptron training rule

1 Begin with random weights w
2 **repeat**
3 **for** *each misclassified example* **do**
4 $w_i = w_i + \eta(t - o)x_i$
5 **until** *all training examples are correctly classified*;
6 **return** w



- **Why would this strategy converge?**
  - Weight does not change when classification is correct

Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 2: Perceptron training rule

1 Begin with random weights w
2 **repeat**
3 | **for** *each misclassified example* **do**
4 | | $w_i = w_i + \eta(t - o)x_i$
5 **until** *all training examples are correctly classified*;
6 **return** w

- **Why would this strategy converge?**
  - Weight does not change when classification is correct
  - If perceptron outputs -1 when target is +1:

Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 2: Perceptron training rule

```
1 Begin with random weights w
2 repeat
3     for each misclassified example do
4         w_i = w_i + η(t − o)x_i
5 until all training examples are correctly classified;
6 return w
```

- **Why would this strategy converge?**
  - Weight does not change when classification is correct
  - If perceptron outputs -1 when target is +1: weight increases ↑

Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 2: Perceptron training rule

1 Begin with random weights w
2 **repeat**
3 **for** *each misclassified example* **do**
4 $w_i = w_i + \eta(t - o)x_i$
5 **until** *all training examples are correctly classified*;
6 **return** w

- **Why would this strategy converge?**
  - Weight does not change when classification is correct
  - If perceptron outputs -1 when target is +1: weight increases ↑
  - If perceptron outputs +1 when target is -1:

Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 2: Perceptron training rule

```
1 Begin with random weights w
2 repeat
3     for each misclassified example do
4         w_i = w_i + η(t − o)x_i
5 until all training examples are correctly classified;
6 return w
```

- **Why would this strategy converge?**
  - Weight does not change when classification is correct
  - If perceptron outputs -1 when target is +1: weight increases ↑
  - If perceptron outputs +1 when target is -1: weight decreases ↓

Perceptron Training Rule

Different algorithms may converge to different acceptable hypotheses

Algorithm 2: Perceptron training rule

```
1 Begin with random weights w
2 repeat
3     for each misclassified example do
4         w_i = w_i + η(t − o)x_i
5 until all training examples are correctly classified;
6 return w
```

- **Why would this strategy converge?**
  - Weight does not change when classification is correct
  - If perceptron outputs -1 when target is +1: weight increases ↑
  - If perceptron outputs +1 when target is -1: weight decreases ↓

Conversion with perceptron training rule is subject to linear separability of training example and appropriate η

Example

Consider the same data

X_1	x_2	Y
1	9	Green
10	9	Green
4	7	Green
4	5	Red
5	3	Red
8	9	Green
4	2	Red
2	5	Red
7	1	Red
2	10	Green
8	5	Green
1	2	Red
8	2	Red

Example

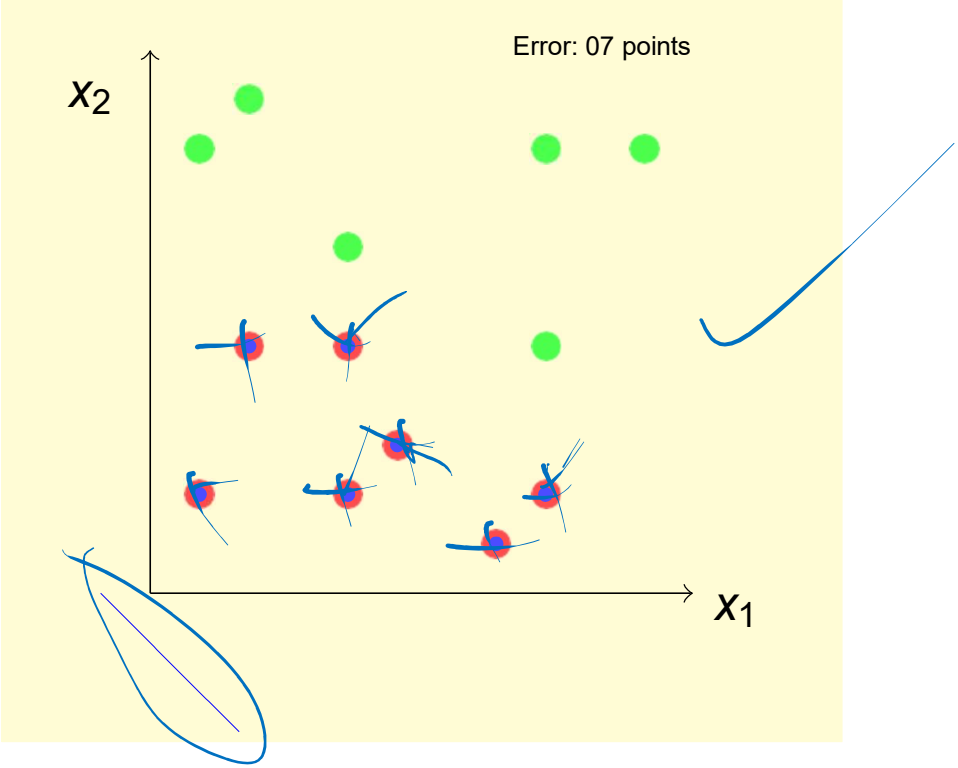
Consider the same data

X_1	x_2	Y
1	9	Green
10	9	Green
4	7	Green
4	5	Red
5	3	Red
8	9	Green
4	2	Red
2	5	Red
7	1	Red
2	10	Green
8	5	Green
1	2	Red
8	2	Red

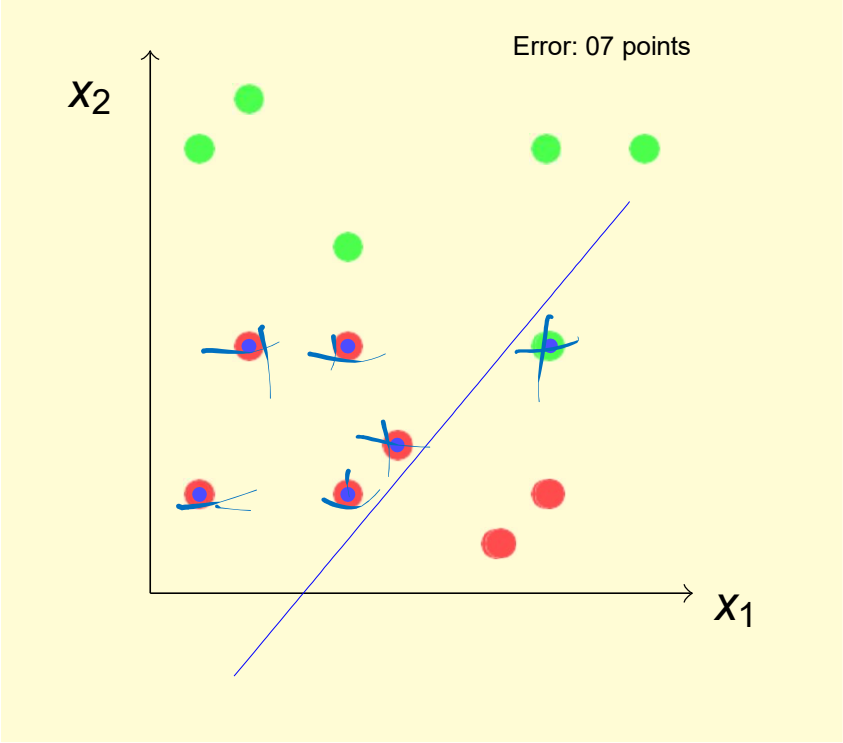
η= 0.01

w0=0.500, w1=0.500, w2=0.500	err=7
w0=0.360, w1=-0.120, w2=0.100	err=6
w0=0.300, w1=-0.180, w2=0.060	err=5
w0=0.240, w1=-0.140, w2=0.140	err=4
w0=0.180, w1=-0.200, w2=0.100	err=5
w0=0.120, w1=-0.160, w2=0.180	err=4
w0=0.080, w1=-0.060, w2=0.180	err=5
w0=0.020, w1=-0.120, w2=0.140	err=4
w0=-0.040, w1=-0.180, w2=0.100	err=5
w0=-0.100, w1=-0.140, w2=0.180	err=4
w0=-0.140, w1=-0.040, w2=0.180	err=5
w0=-0.200, w1=-0.100, w2=0.140	err=3
w0=-0.260, w1=-0.160, w2=0.100	err=4
w0=-0.320, w1=-0.120, w2=0.180	err=3
w0=-0.360, w1=-0.020, w2=0.180	err=3
w0=-0.420, w1=-0.080, w2=0.140	err=2
w0=-0.420, w1=-0.080, w2=0.240	err=2
Fourteen more iterations	
w0=-0.900, w1=-0.020, w2=0.180	err=1
w0=-0.900, w1=-0.020, w2=0.240	err=2
w0=-0.920, w1=0.020, w2=0.220	err=2
w0=-0.960, w1=-0.020, w2=0.220	err=3
w0=-0.980, w1=0.020, w2=0.200	err=2
w0=-1.000, w1=0.060, w2=0.180	err=2
w0=-1.040, w1=0.020, w2=0.180	err=0

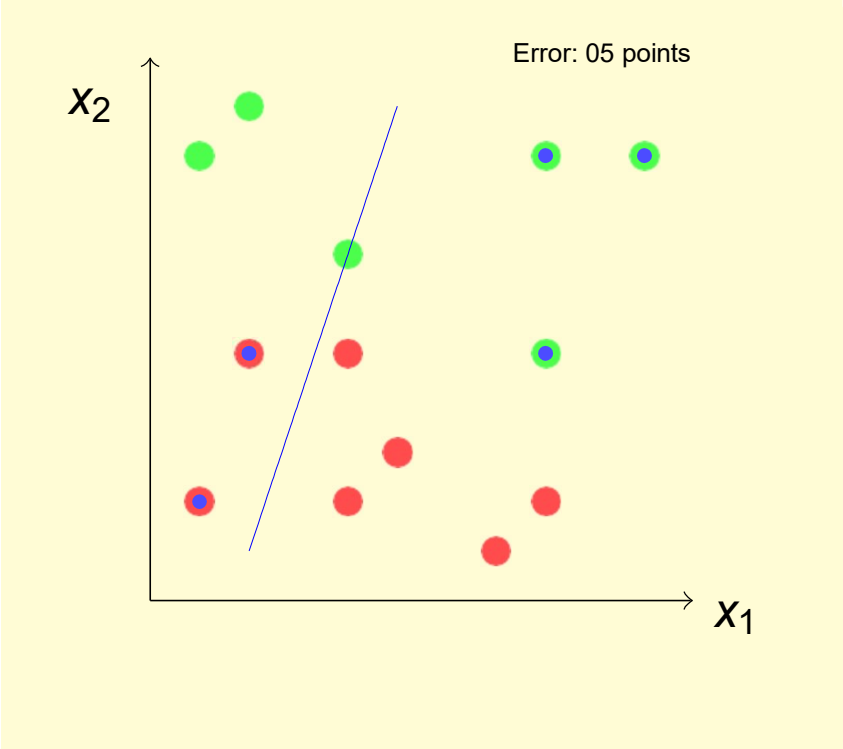
Visual Interpretation



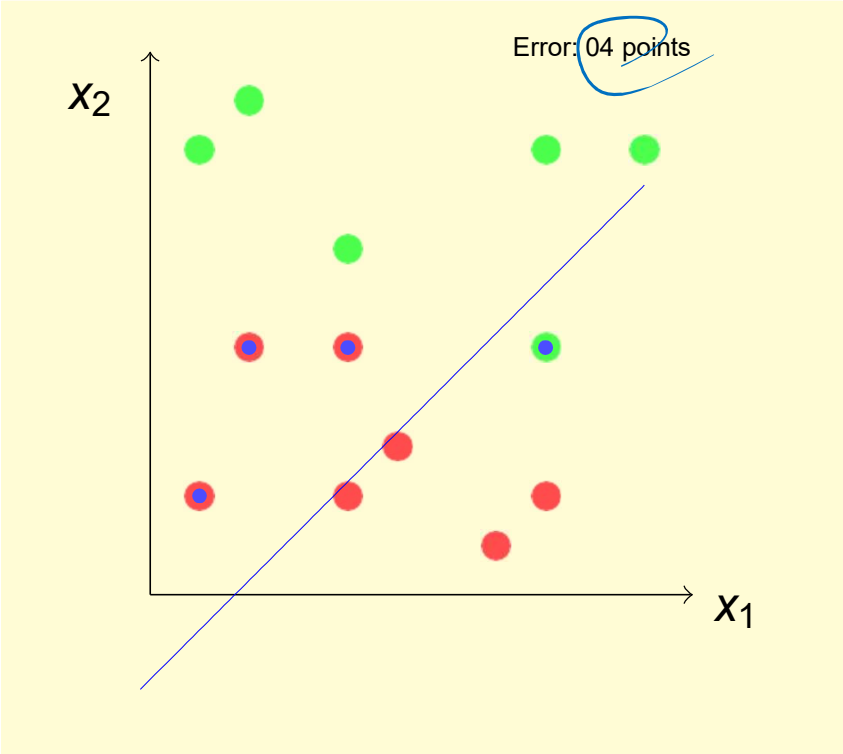
Visual Interpretation



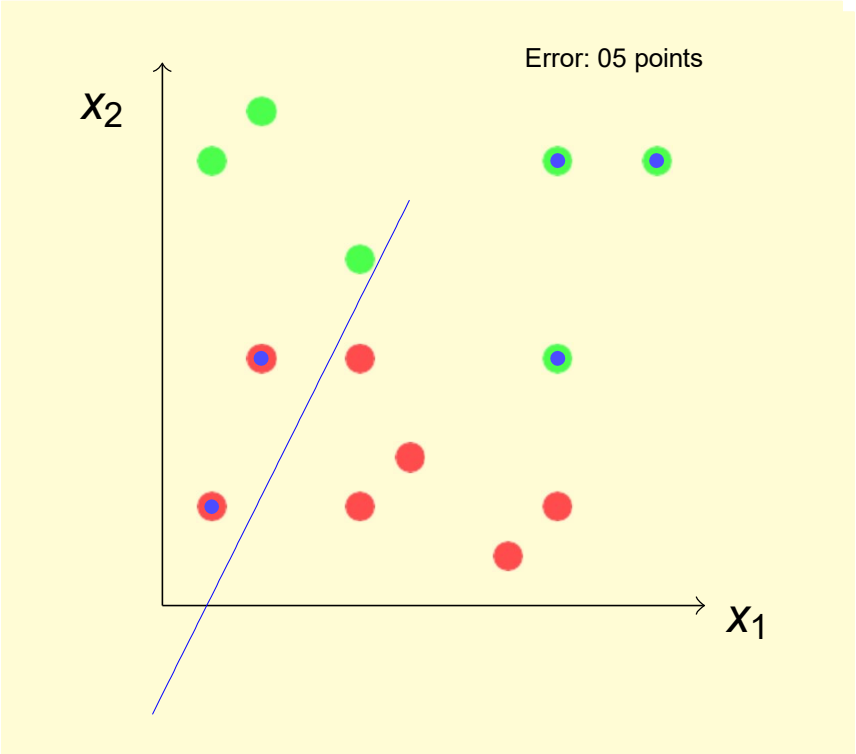
Visual Interpretation



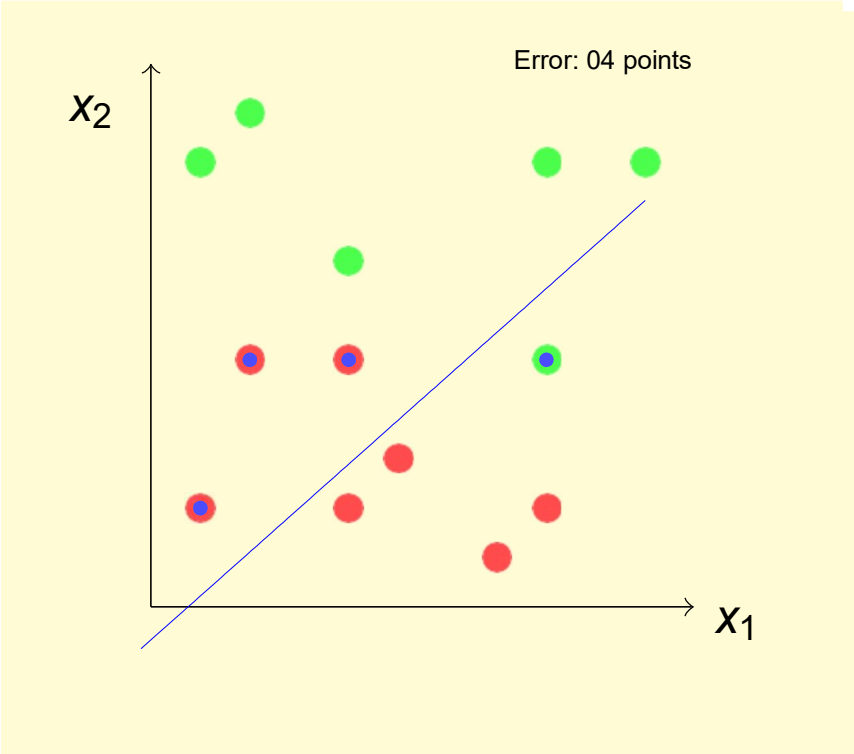
Visual Interpretation



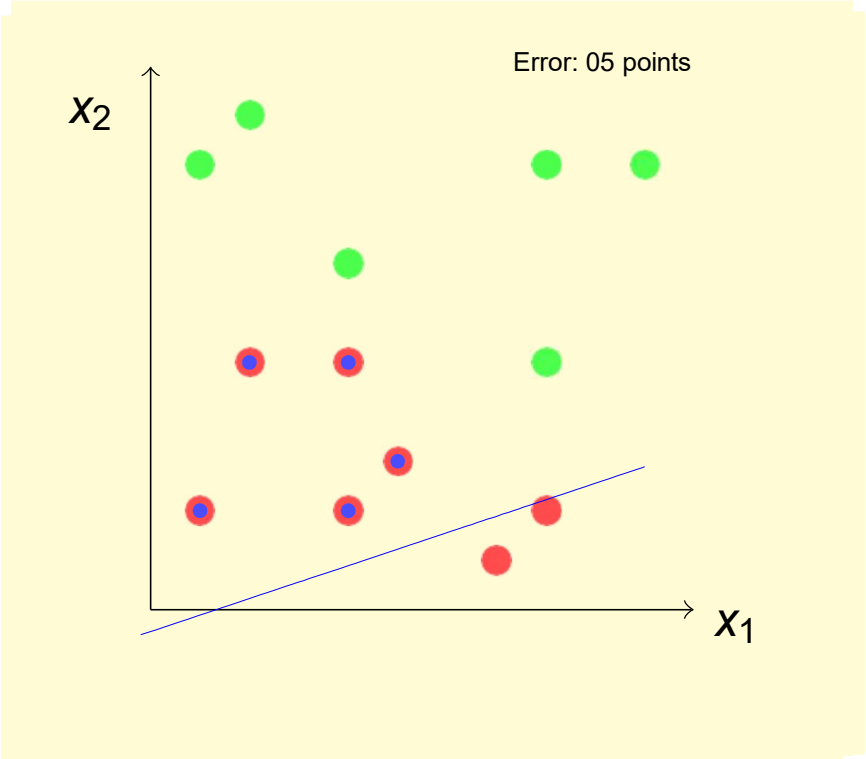
Visual Interpretation



Visual Interpretation

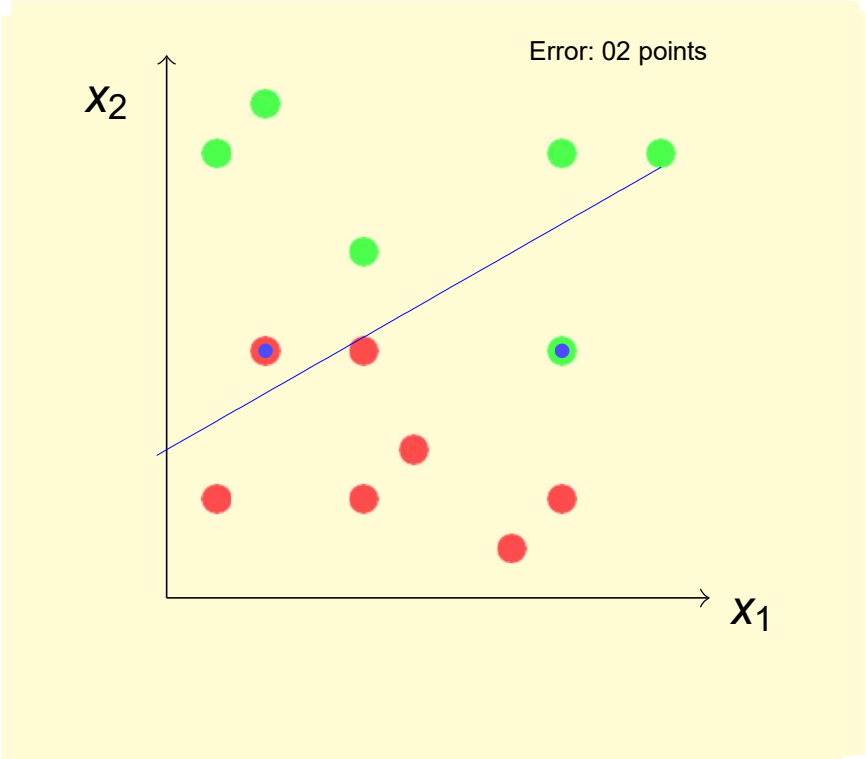


Visual Interpretation

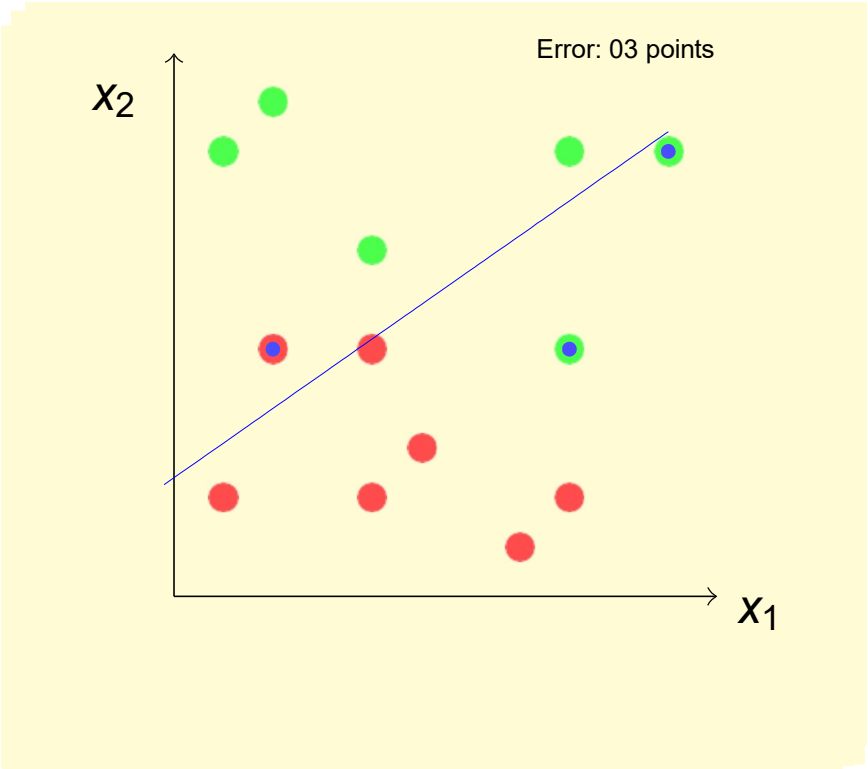


Visual Interpretation

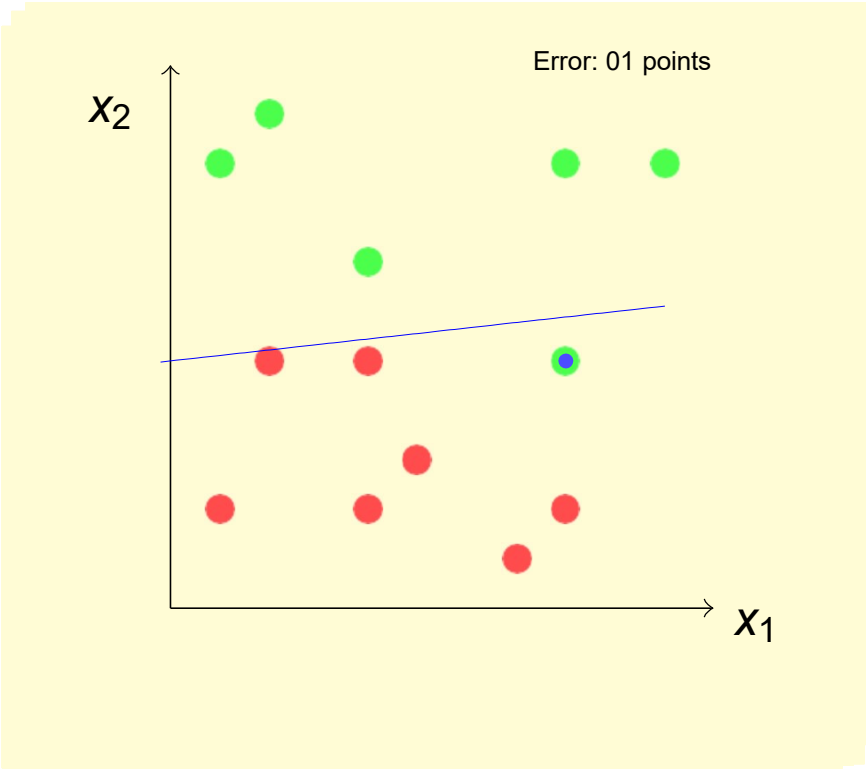
Visual Interpretation



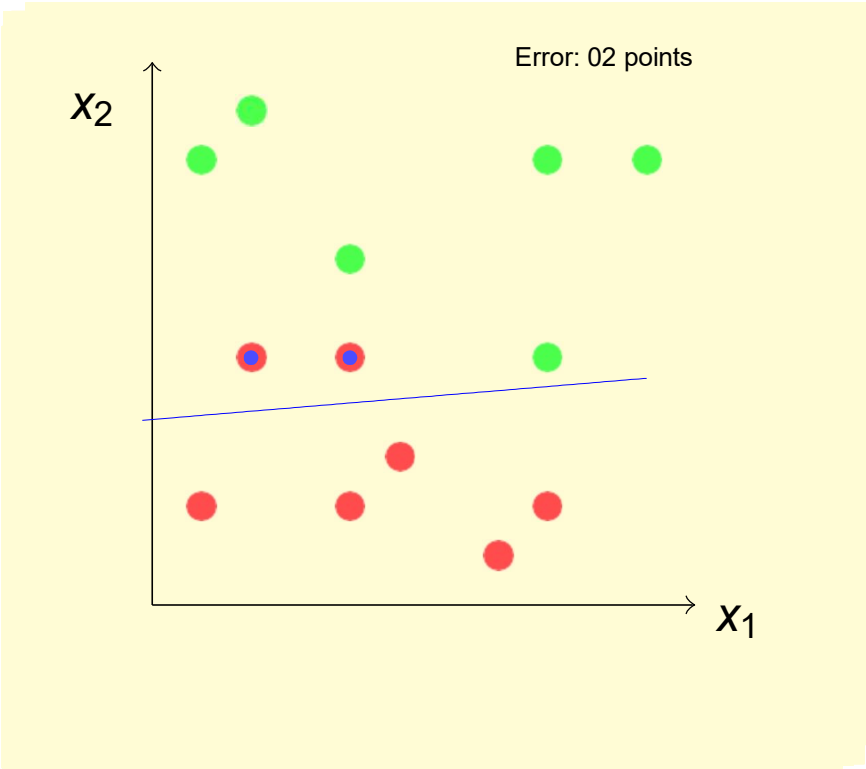
Visual Interpretation



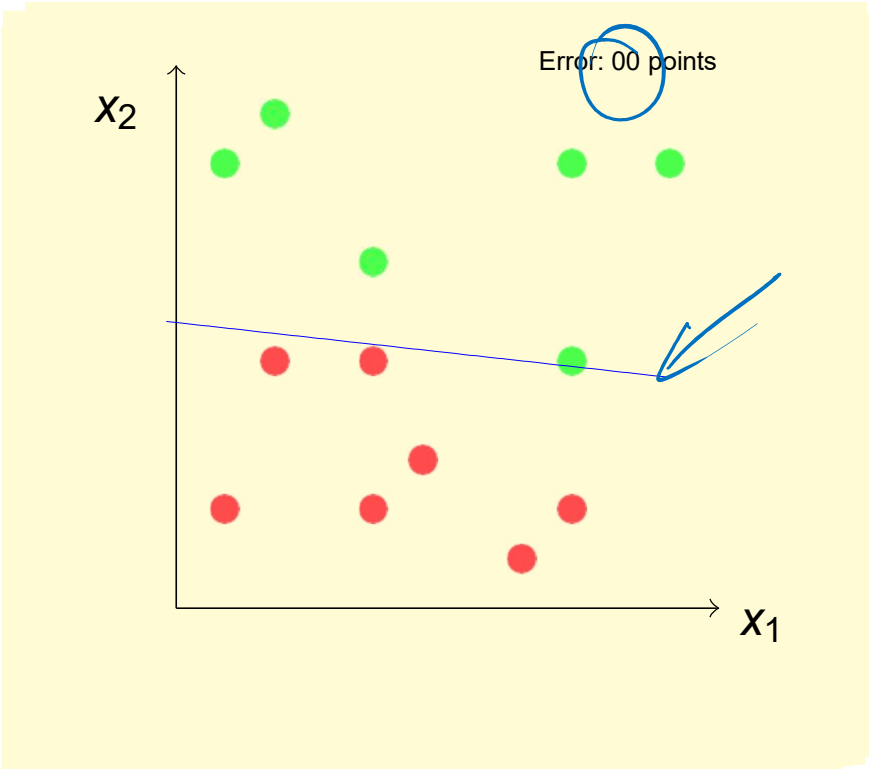
Visual Interpretation



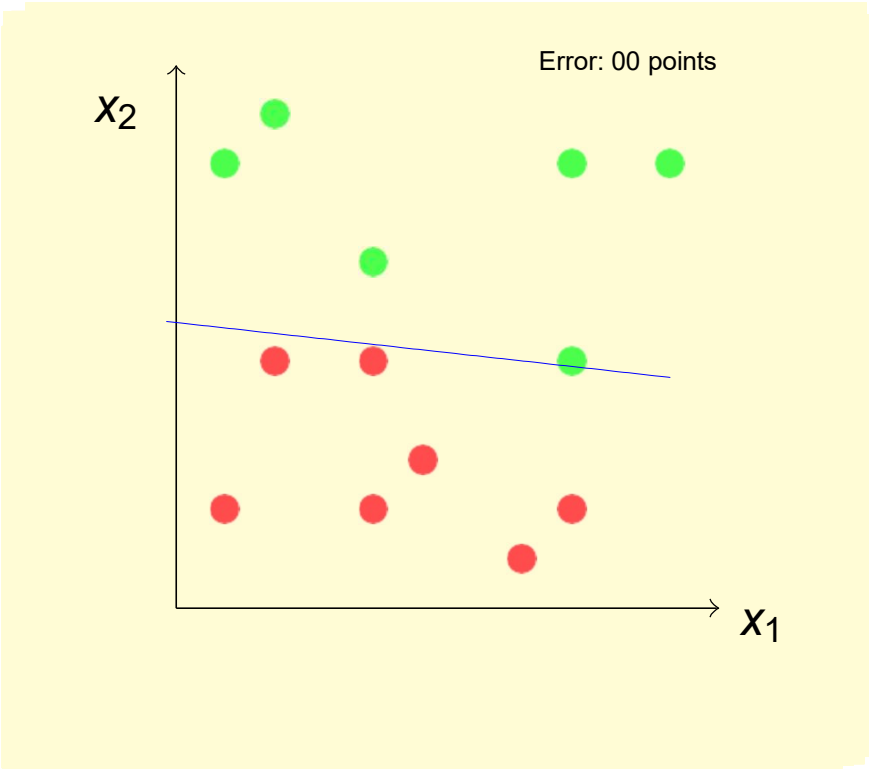
Visual Interpretation



Visual Interpretation



Visual Interpretation



- Conversion is not gradual. (Error is NOT reducing monotonically)
- It is difficult to decide when to stop if data is not linearly separable

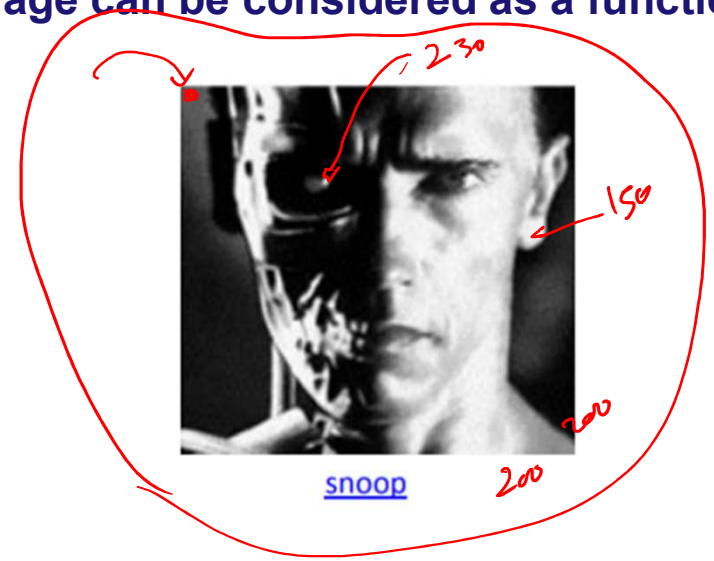
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

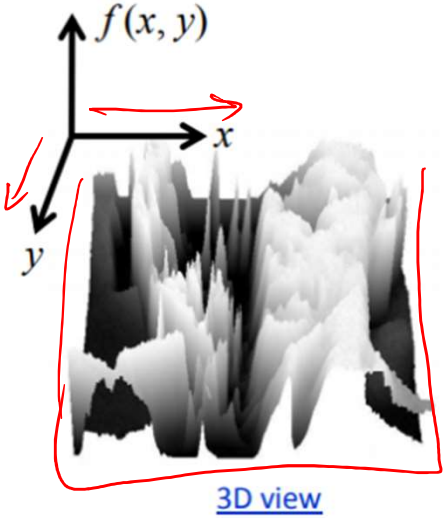
CNN Prerequisites: Computer Vision

Dr. Kamlesh Tiwari

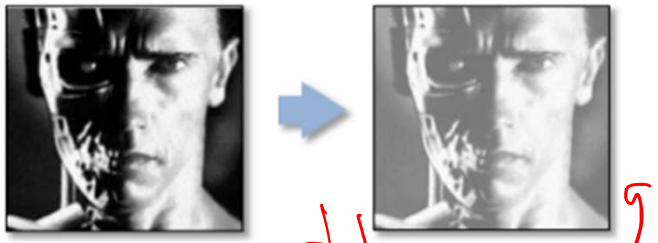
Image

Grayscale image can be considered as a function f : R 2 → R

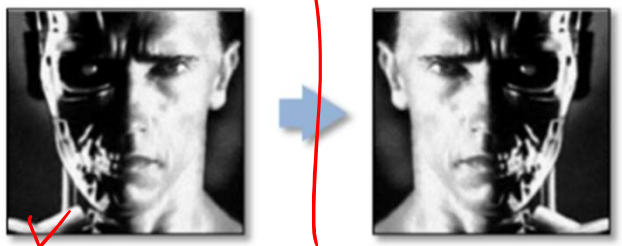




- Being digital adds quantization and sampling. We may apply operators on this function



$g(x,y) = f(x,y) + 20$



$g(x,y) = f(-x,y)$

Computer Vision - history

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group
Vision Memo. No. 100.

July 7, 1966

THE SUMMER VISION PROJECT

Seymour Papert

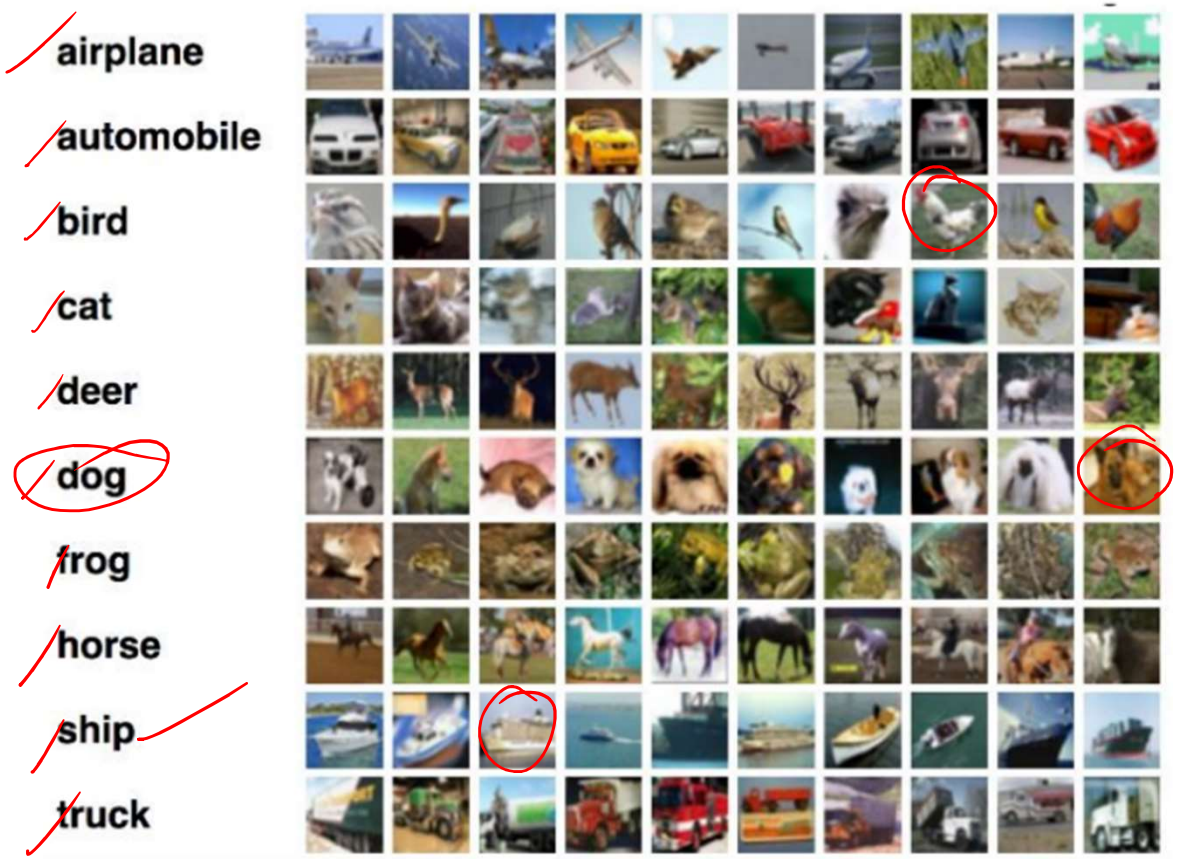
The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

Some Tasks

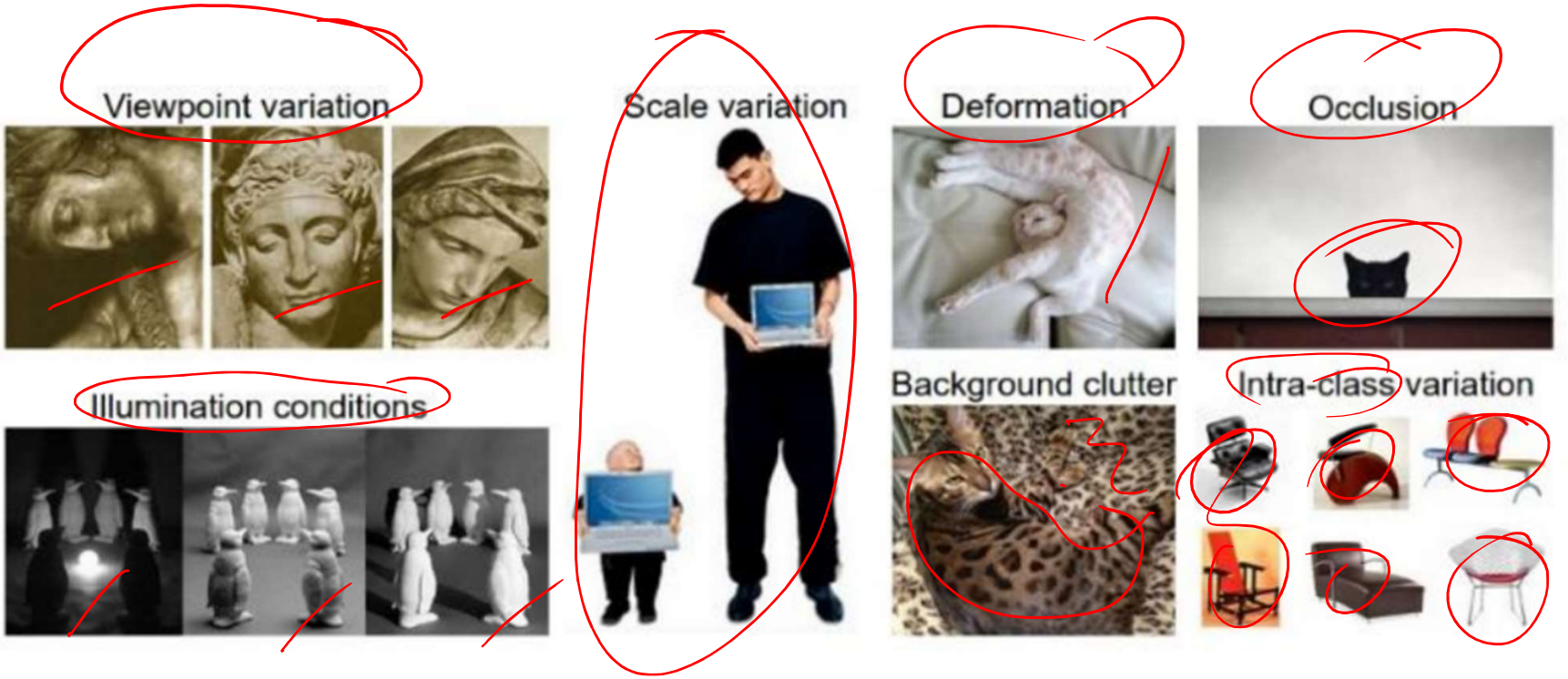
- Classification
- Annotation
- Detection
- Segmentation

How the Afghan Girl was Identified by her Iris Patterns 1984-2002

Classification



Classification Challenges



Annotation

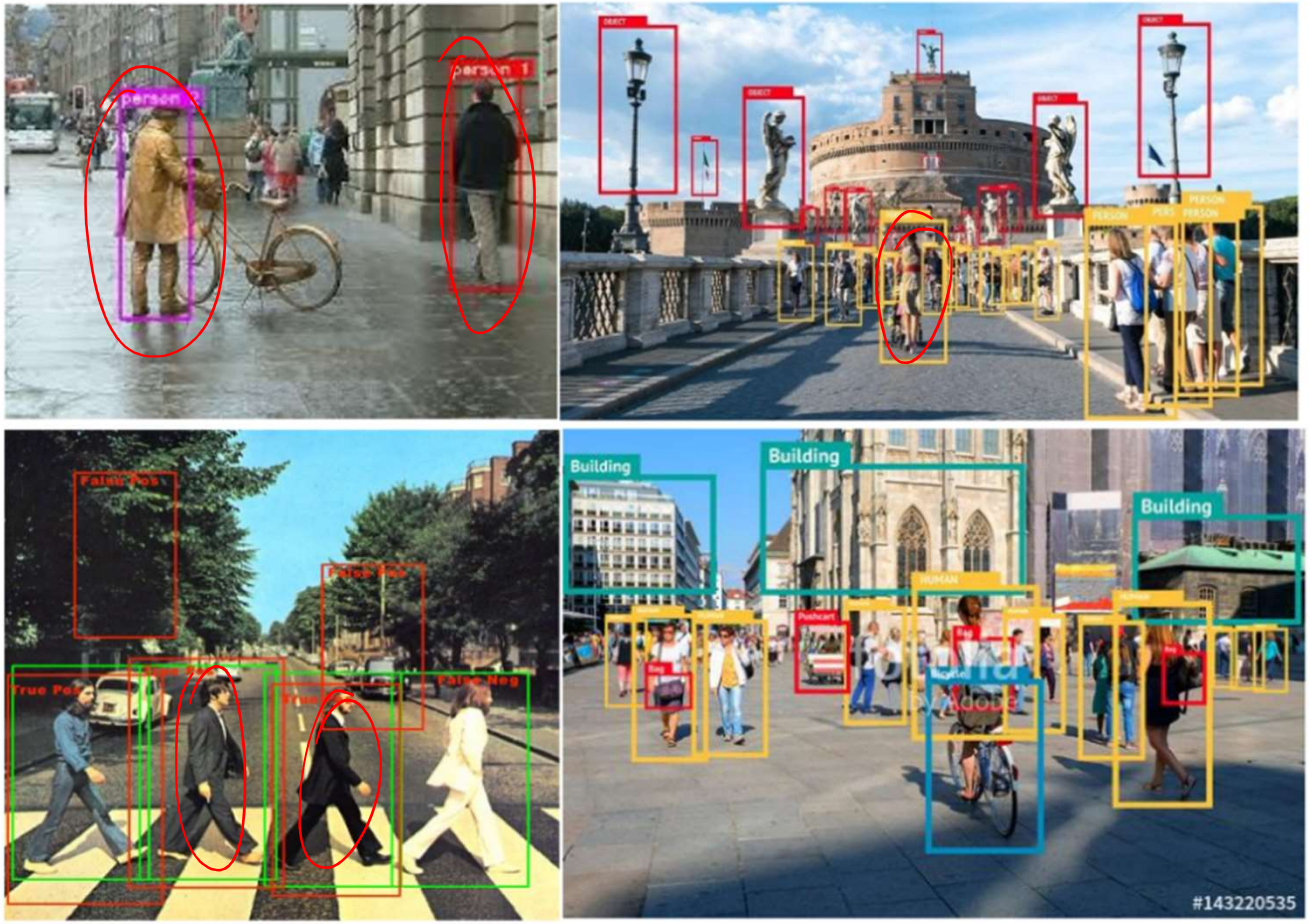
					
Predicted keywords	sky, jet, plane, smoke, formation	grass, rocks, sand, valley, canyon	sun, water, sea, waves, birds	water, tree, grass, deer, white-tailed	bear, snow, wood, deer, white-tailed
Human annotation	sky, jet, plane, smoke	rocks,sand, valley, canyon	sun, water, clouds, birds	tree, forest, deer, white-tailed	tree, snow, wood, fox

Prediction for queries: Sky (Row1), Street (Row2), Mare (Row3) and Train (Row4)

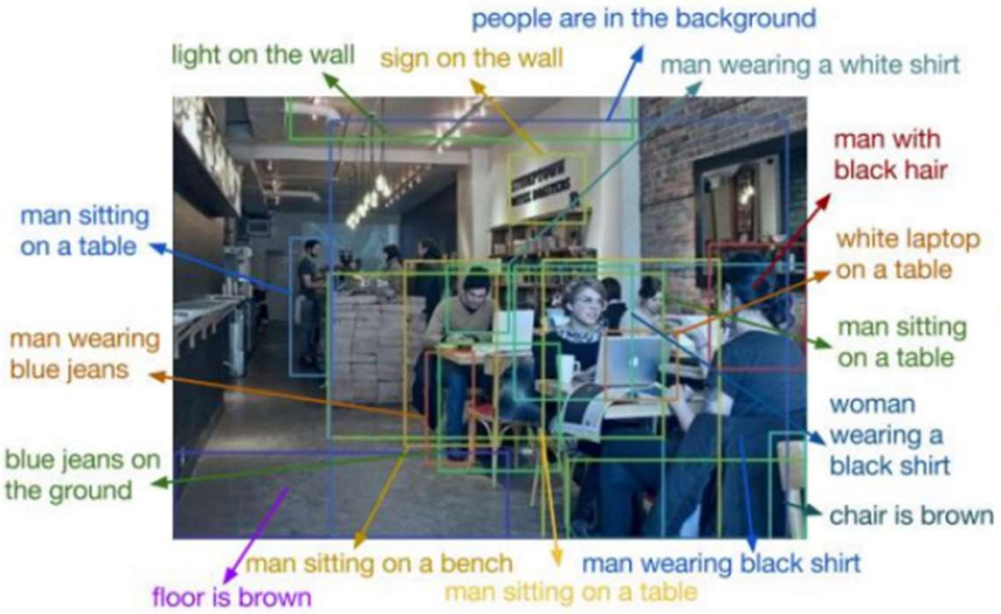
Ref: A. Makadia, V. Pavlovic, and S. Kumar.

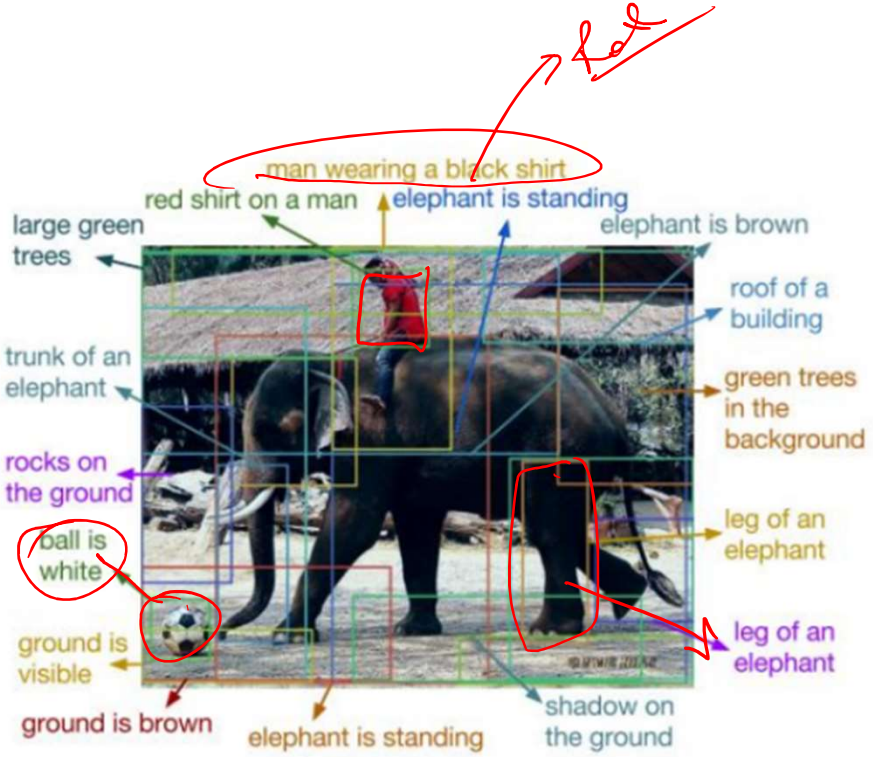
A New Baseline for Image Annotation. In Proceedings of ECCV 08.

Detection

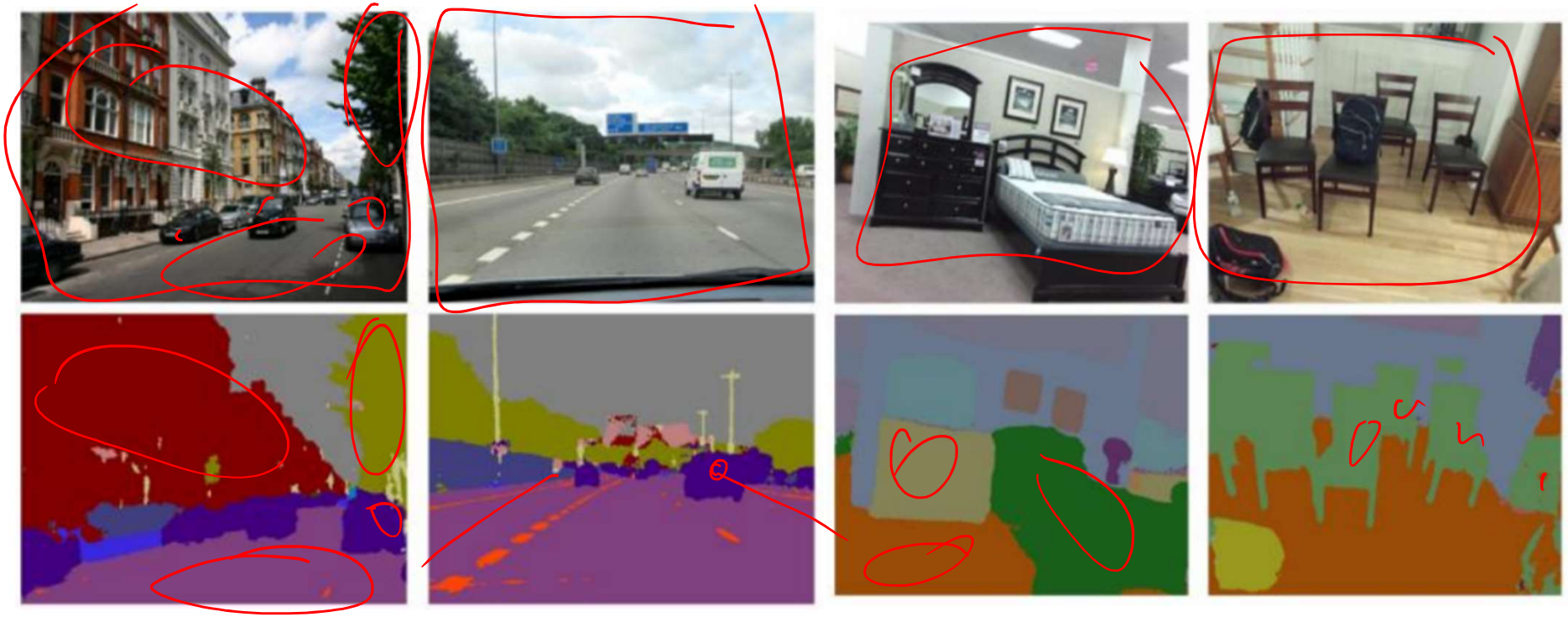


Description



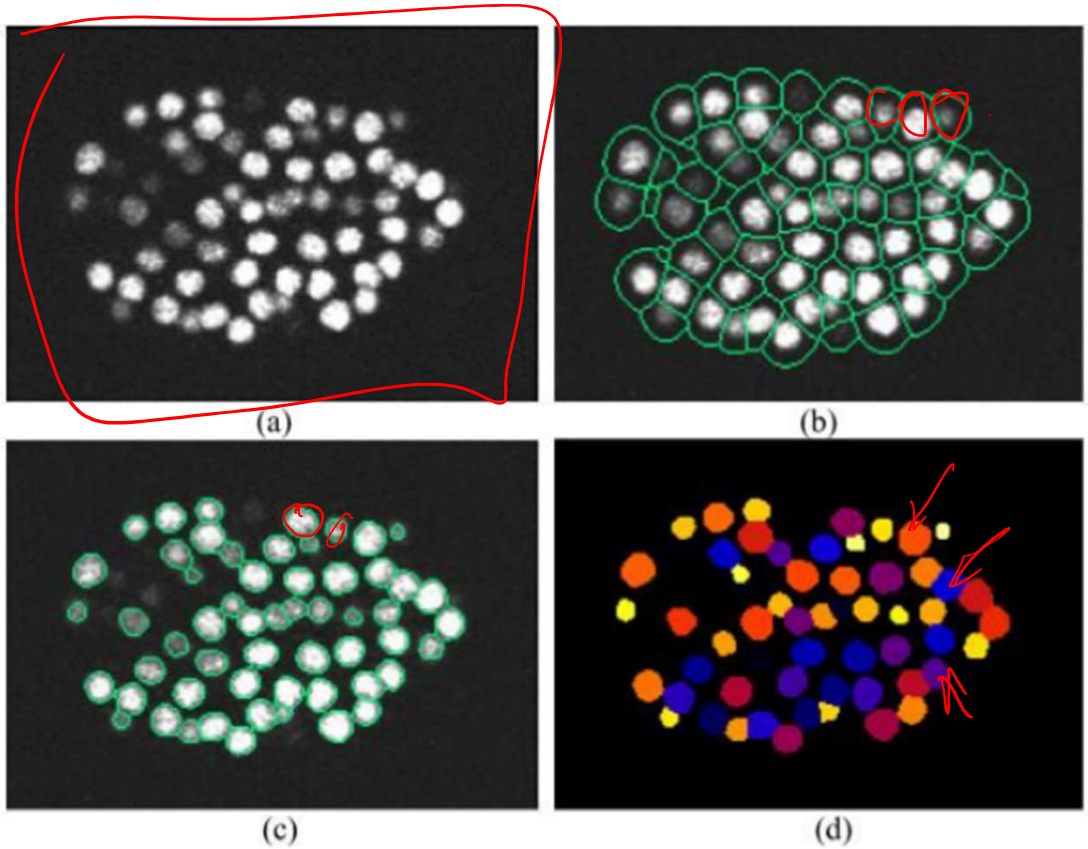


Segmentation



Vijay Badrinarayanan, Alex Kendall and Roberto Cipolla "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." PAMI, 2017.

Segmentation



Thank You!

In our next session: Image Operations

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CNN Prerequisites: Computer Vision

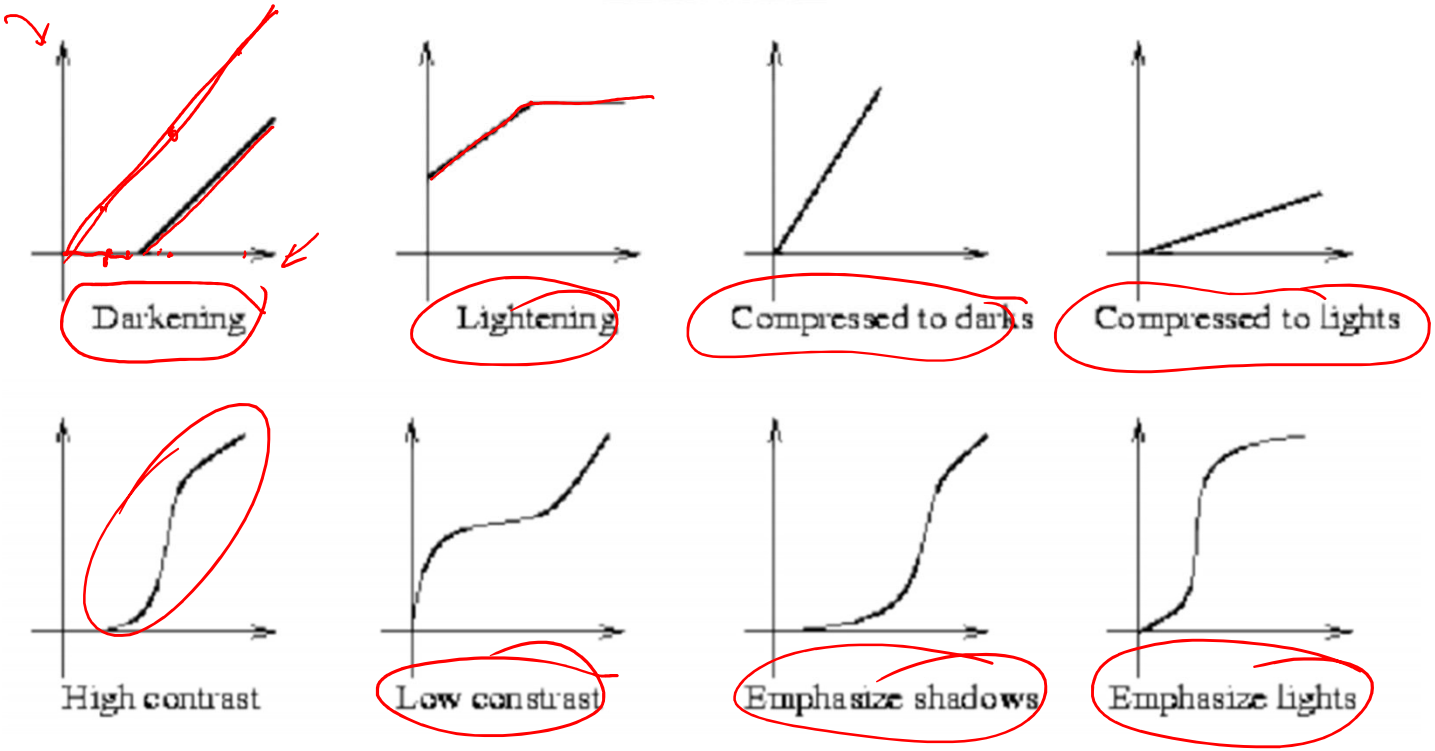
Dr. Kamlesh Tiwari

Point Operations on Images

Point operation is defined as

$$s = M(r)$$

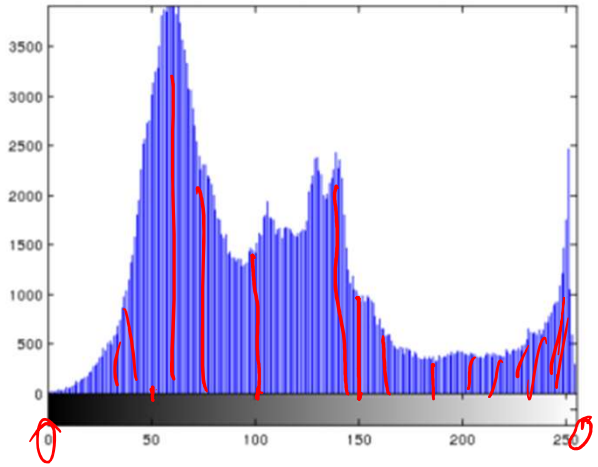
where r is source pixel intensity and s is destination pizel intensity



Histogram

- Discrete probability distribution of the image intensity values

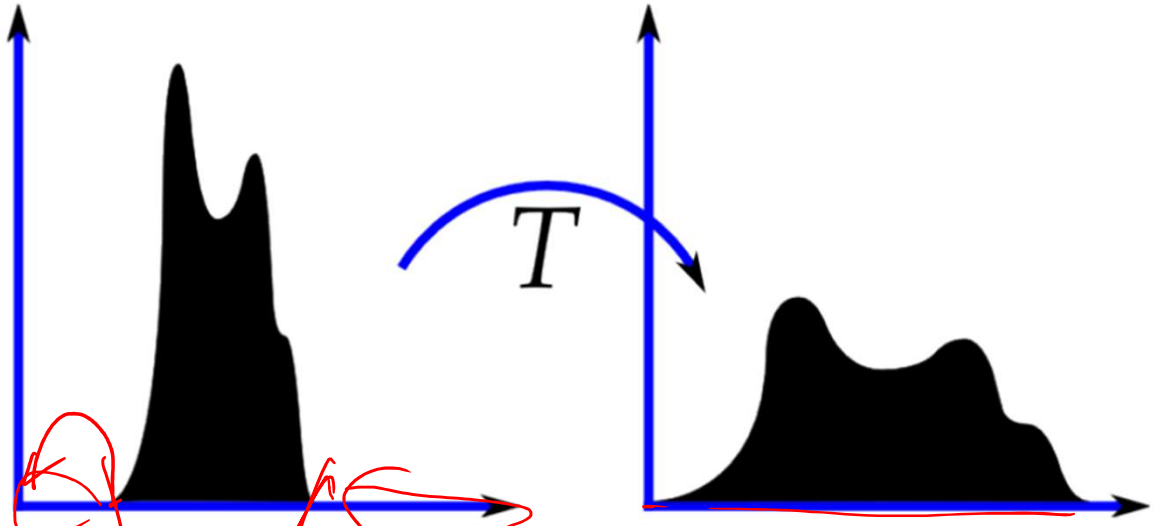
(a) Image



(b) Histogram

Histogram Equalization

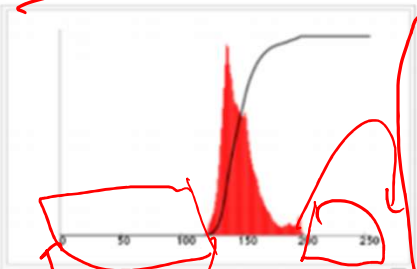
One method to enhance images is to equalize the histogram



Exercise: Prove that an image transformed by its cumulative distribution function results in an image with uniform histogram. More generally, histogram can be modified by histogram specification

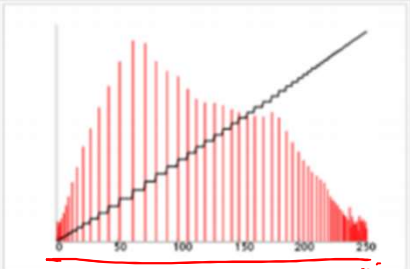
Histogram Equalization Examples

An unequalized image

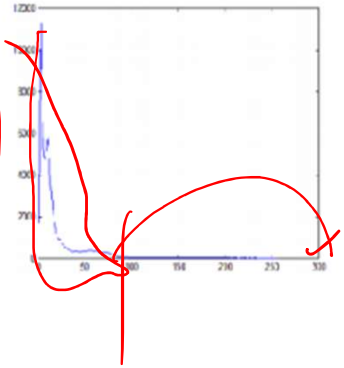


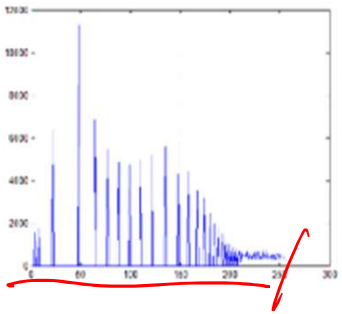
Corresponding histogram (red) and cumulative histogram (black)

The same image after histogram equalization



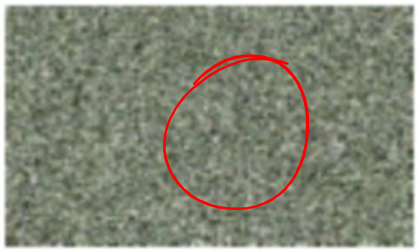
Corresponding histogram (red) and cumulative histogram (black)

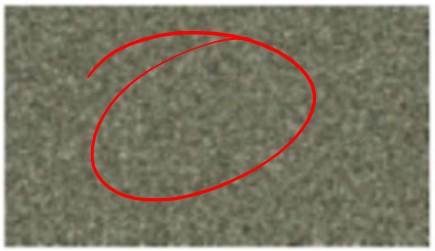




Shuffling the pixels

- What happens if we shuffle all the pixels in an image randomly?

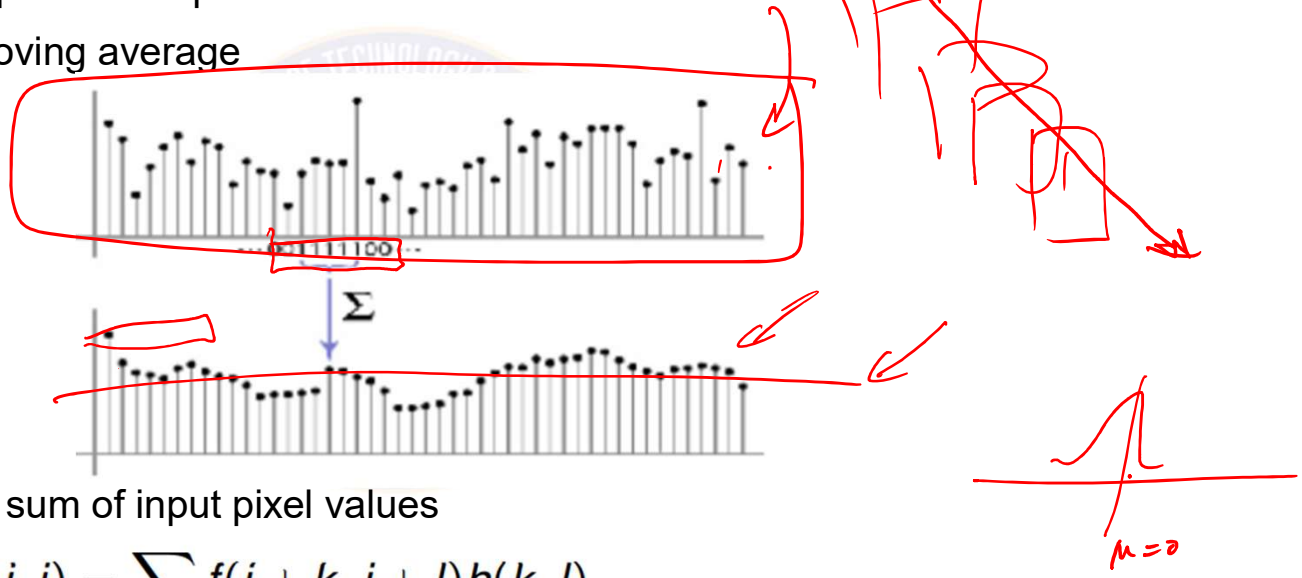





- Different images may have same histogram
- Need more local operators

First attempt at a solution

- Replace each pixel with an average of all the values in its neighbourhood (assumptions; a) pixels are like neighbour 2) noise is independent of pixel location
- Can add weights to our moving average



- Output pixel is a weighted sum of input pixel values

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$$

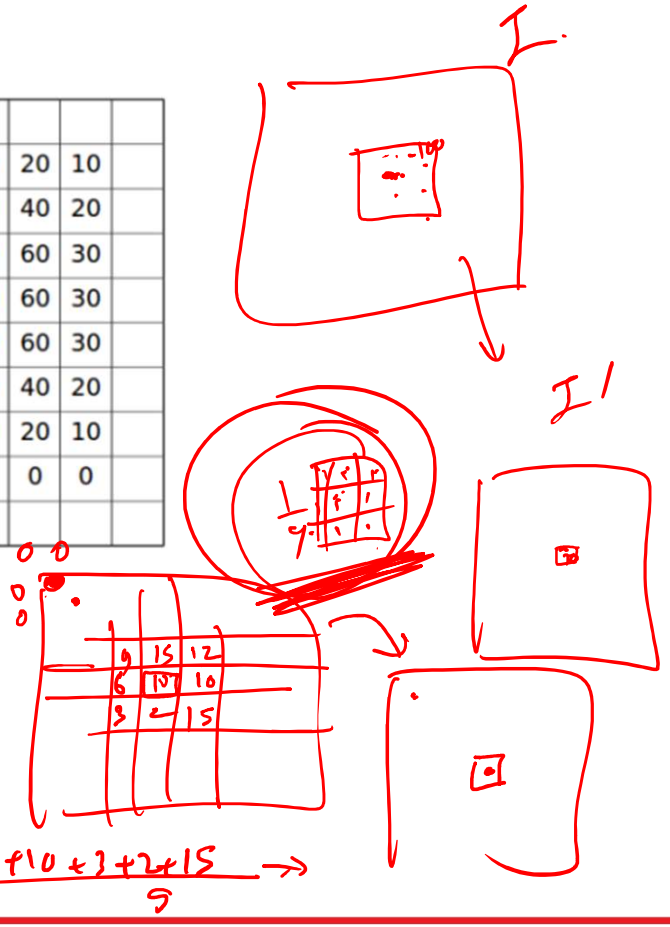
- Entries in kernel h(k, l) are called the filter coefficients. Operator is termed correlation operator. g = f ⊗ h

Convolution

$$g = f \otimes h.$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

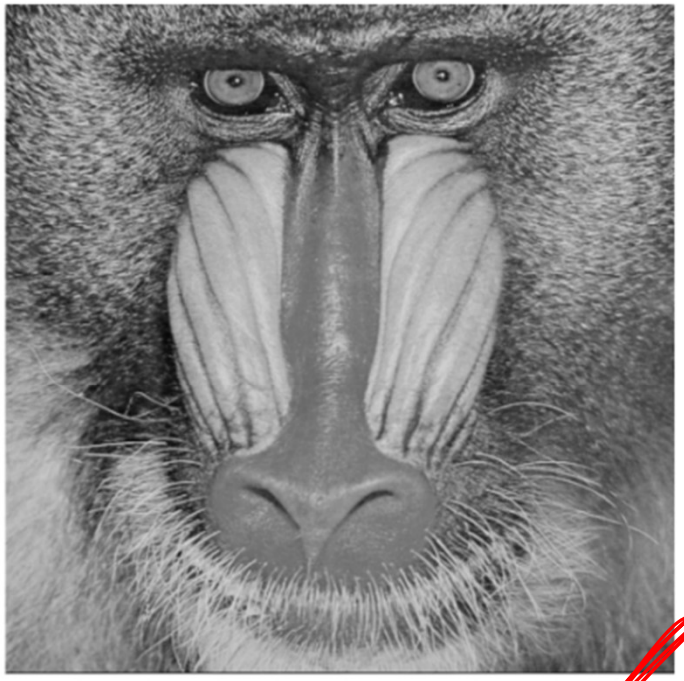
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	



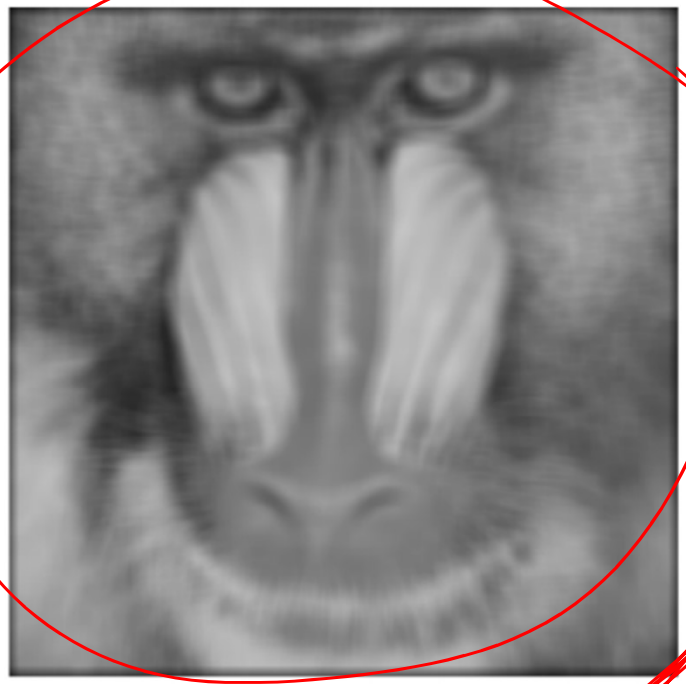
- Convolution is a simple variant of correlation termed as g = f ∗ h

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l)$$

Box Filtering



(c)



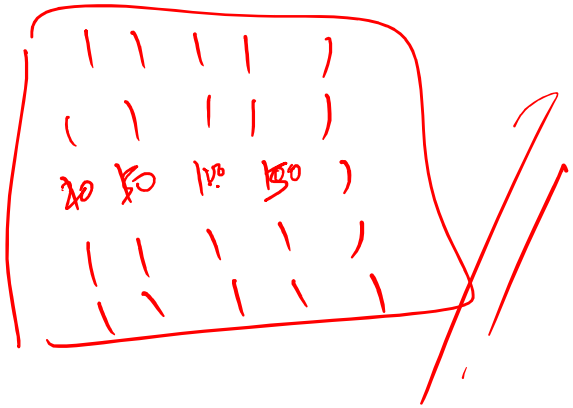
(d)

Gaussian Filtering

Used when we want to have central pixels with more influence.

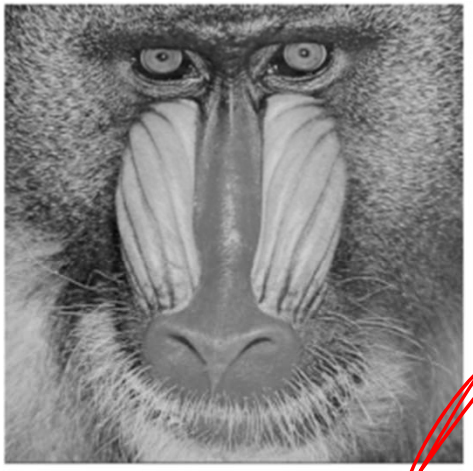
- Gaussian Kernel is defined as

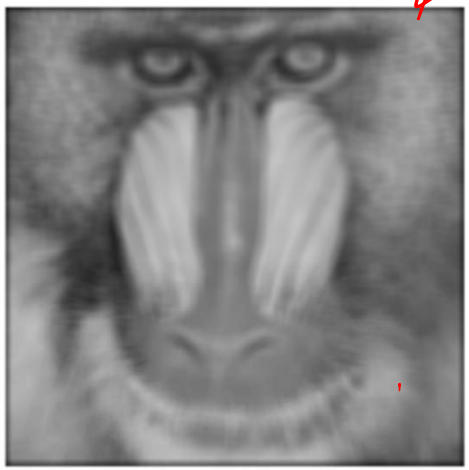
$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



- It is a low pass filter

Gaussian Filtering





Gaussian filtering

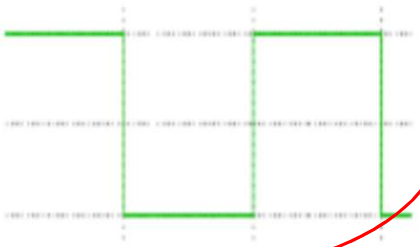
$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \quad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Convolution

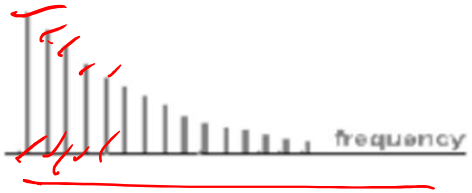
Fourier Transform

Any signal (function) can be written as sum of various sin and cos waves (terms)

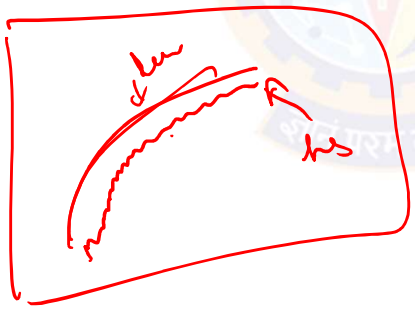
$$A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)$$

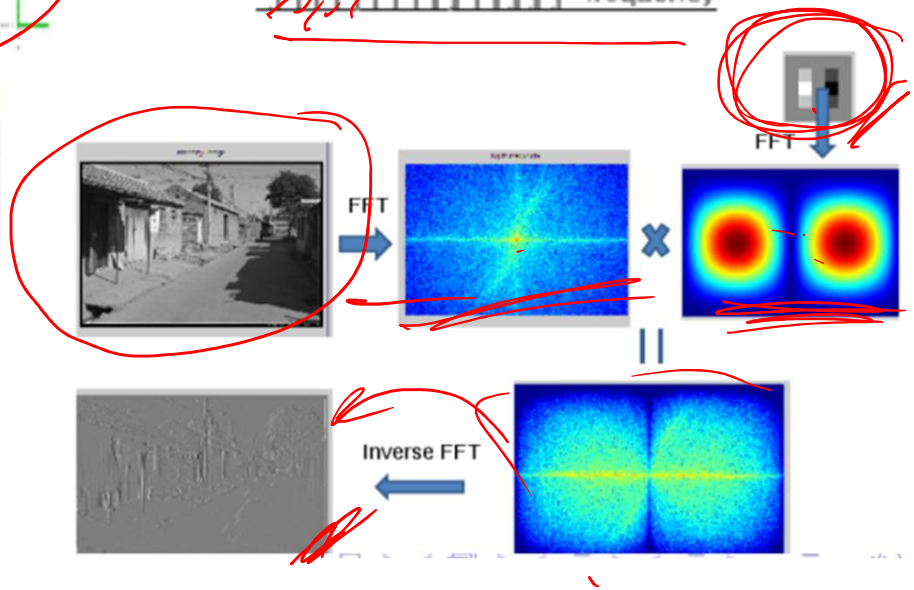


=



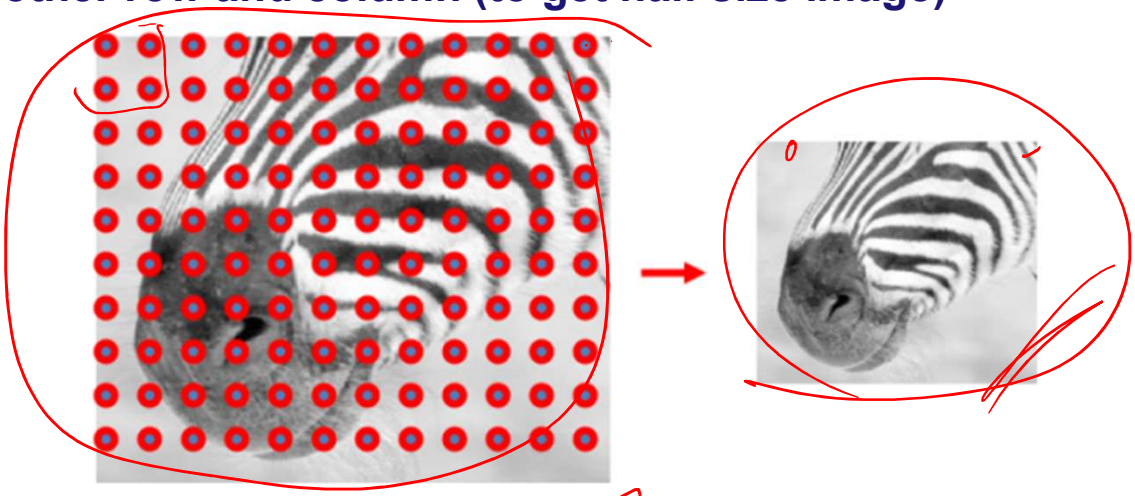
- $F[g \times h] = F[g] \times F[h]$
- $g \times h = F^{-1}[F[g] \times F[h]]$





Sampling

Throw away every other row and column (to get half size image)

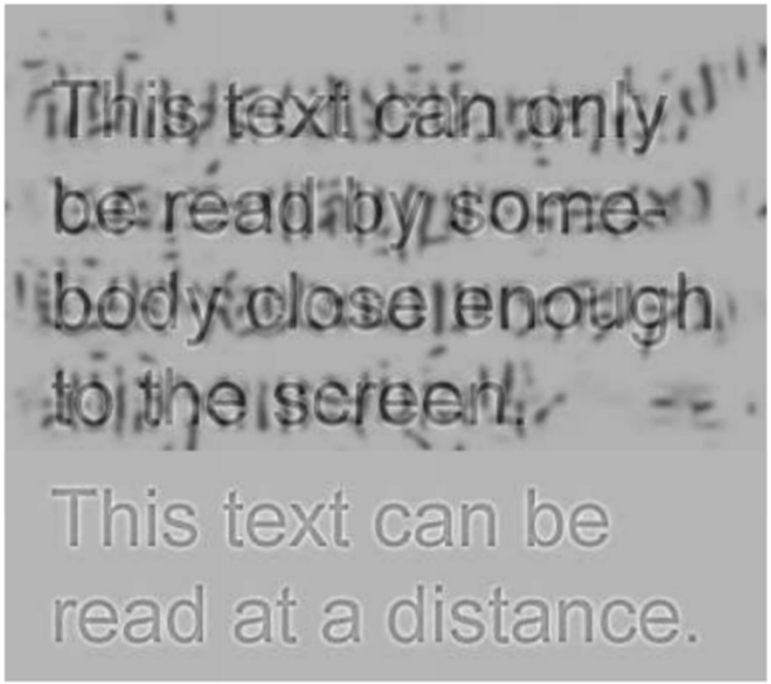


Aliasing Problem

Sub sampling may be dangerous: Funny stripped shirt. Wagon wheels rolling in the wrong way in movies.

Hybrid Images

Example Hybrid Image



The hybrid font becomes invisible at few meters. The bottom text remains easy to read at relatively long distances.

Thank You!

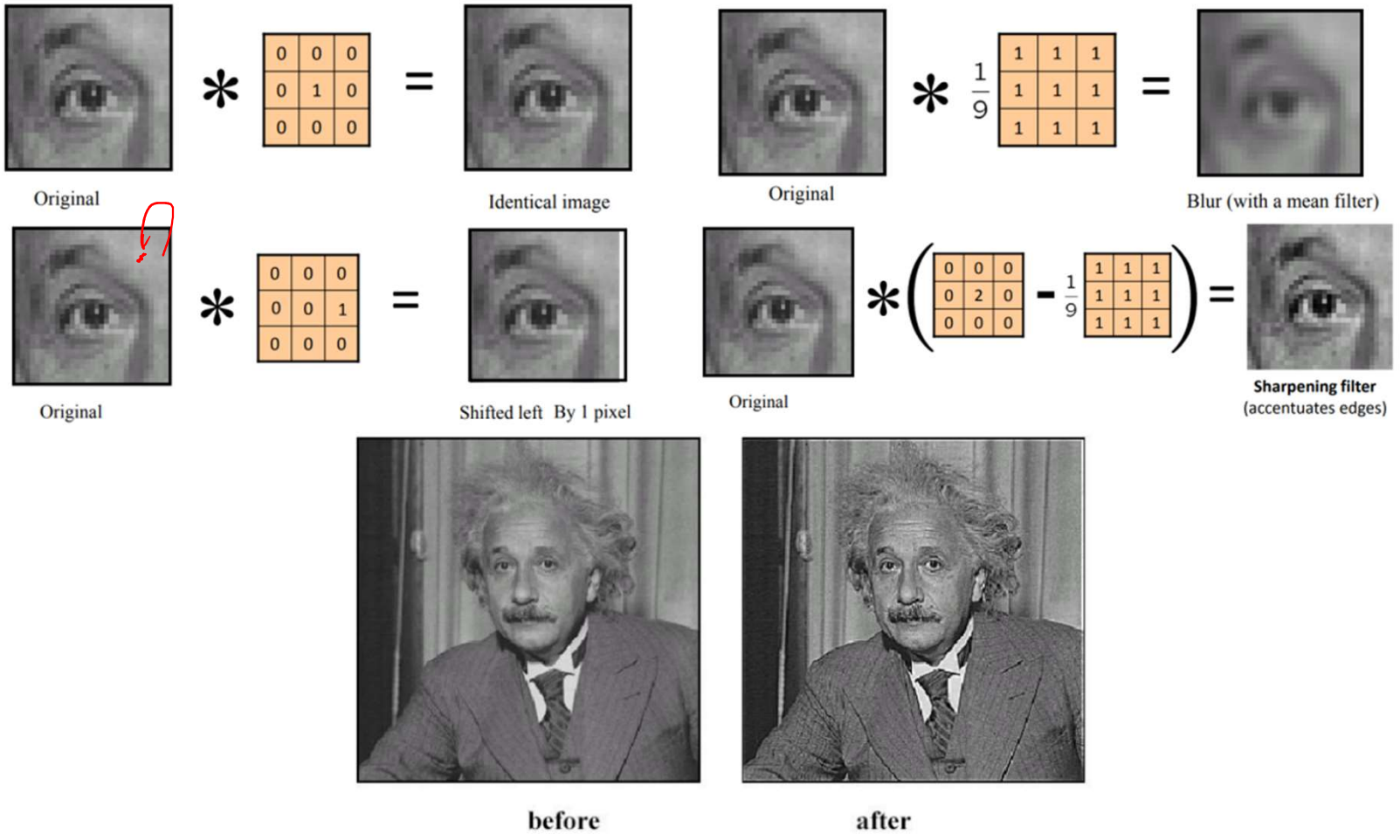
In our next session: Filtering

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CNN Prerequisites: Computer Vision

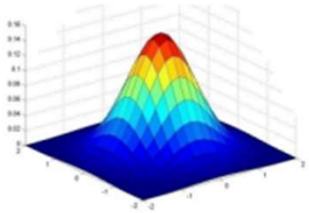
Dr. Kamlesh Tiwari

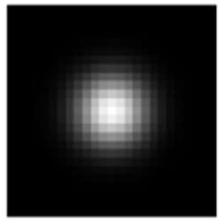
Linear filtering



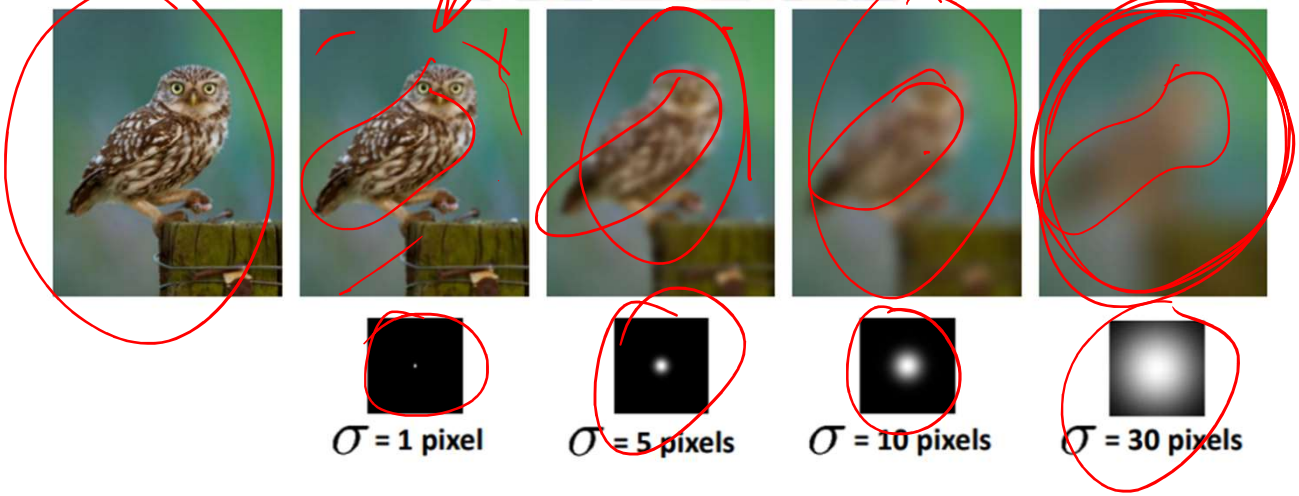
Gaussian filter

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





Act as low-pass filter as it removes high-frequency components. Convolution with self is another Gaussian.



Convolving two times with Gaussian kernel of σ width is equivalent to Convoluting once with Gaussian kernel of $\sqrt{2}\sigma$ width

Sharpening

What does blurring take away?

Let's add it back:

Edge detection

Common approximation of derivative of Gaussian

$\frac{1}{8}$

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

s_x

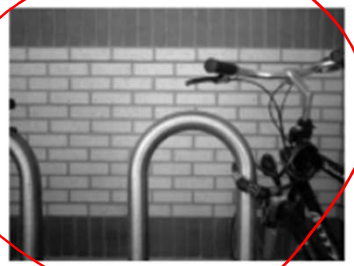
$\frac{1}{8}$

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

s_y

Edge magnitude is $\sqrt{s_x^2 + s_y^2}$ and the direction is $\tan^{-1}(s_y/s_x)$

Edge is detected by using threshold

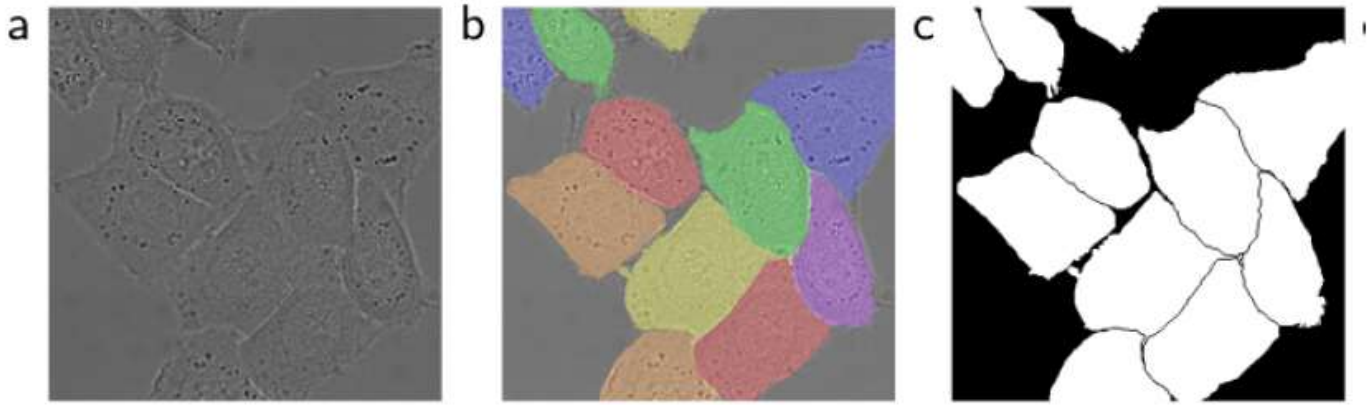



BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CNN Case Studies (U-Net)

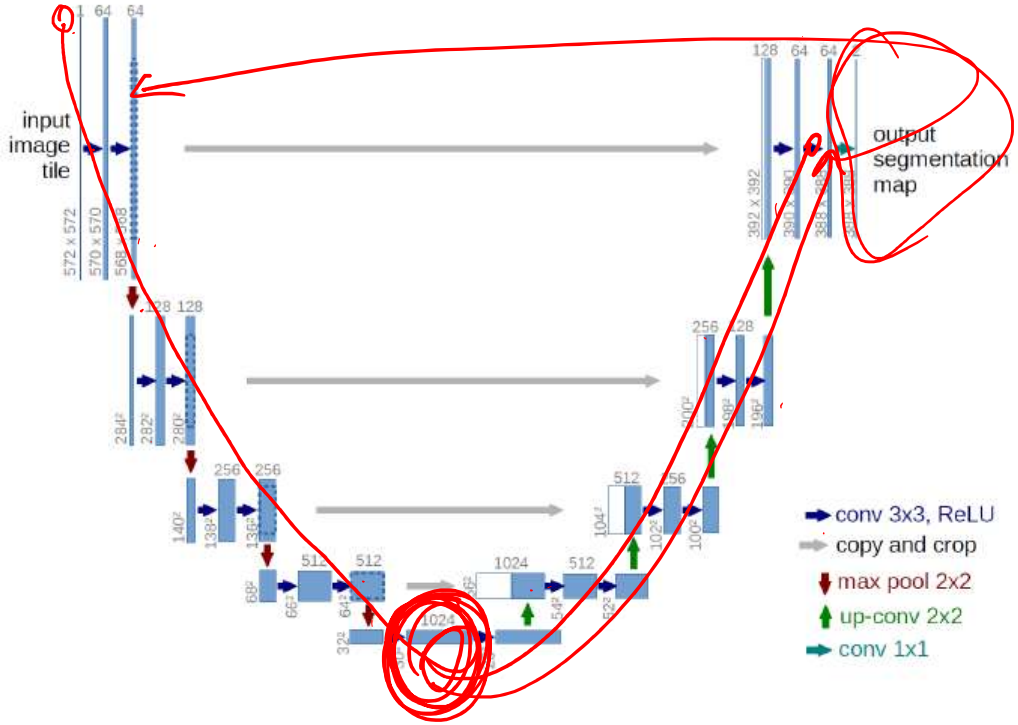
Dr. Kamlesh Tiwari

Segmentation



- ISBI challenge for segmentation of neuronal structures in electron microscopic stacks
- Works with very few training images (30/application) and touching boundary. Yield more precise segmentation
- Data augmentation is essential (mainly shift, rotation and elastic deformation)

U-Net



- ISBI DIC-HeLa achieved 77.6% IoU as compared to 46.0% second
- ISBI Cell tracking 2015, achieved 92% IoU as compared to 83% second

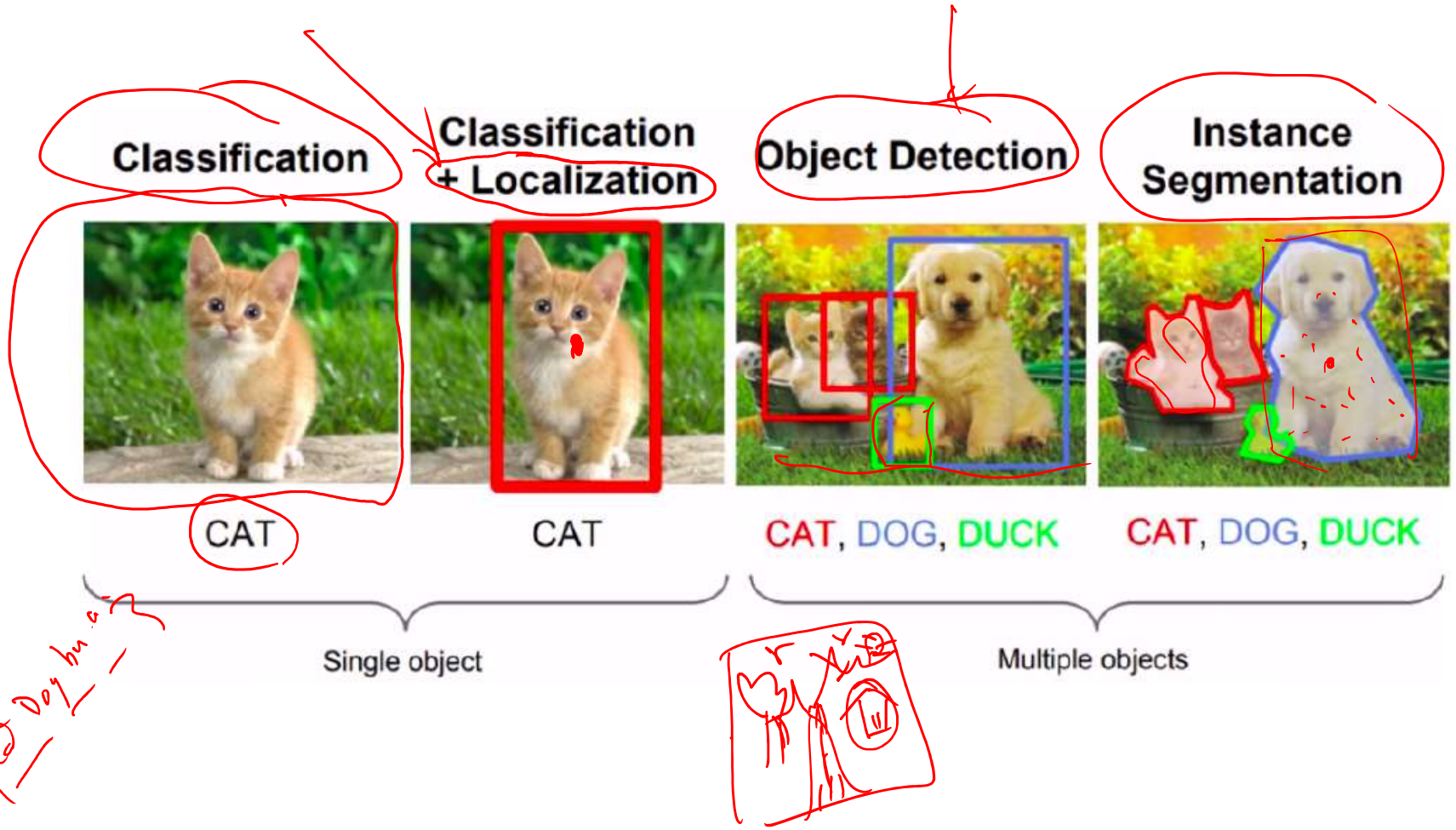
Cite 9576 O. Ronneberger and P.Fischer and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, Medical Image Computing and Computer-Assisted Intervention (MICCAI), LNCS-9351, pages 234–241, Springer-2015

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

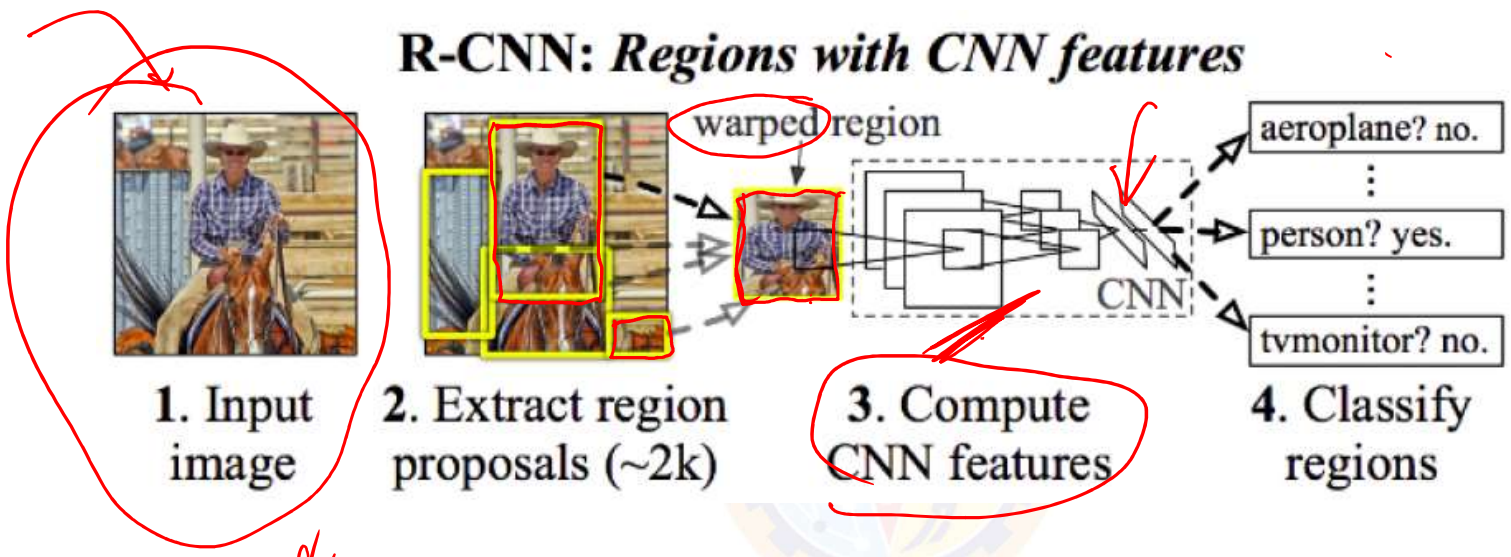
CNN Case Studies (R-CNN)

Dr. Kamlesh Tiwari

Object Detection and Localization



R-CNN



- Region proposals 20K (from external selective search)
- Warped image (resizing)
- SVM for classification (one for each class)

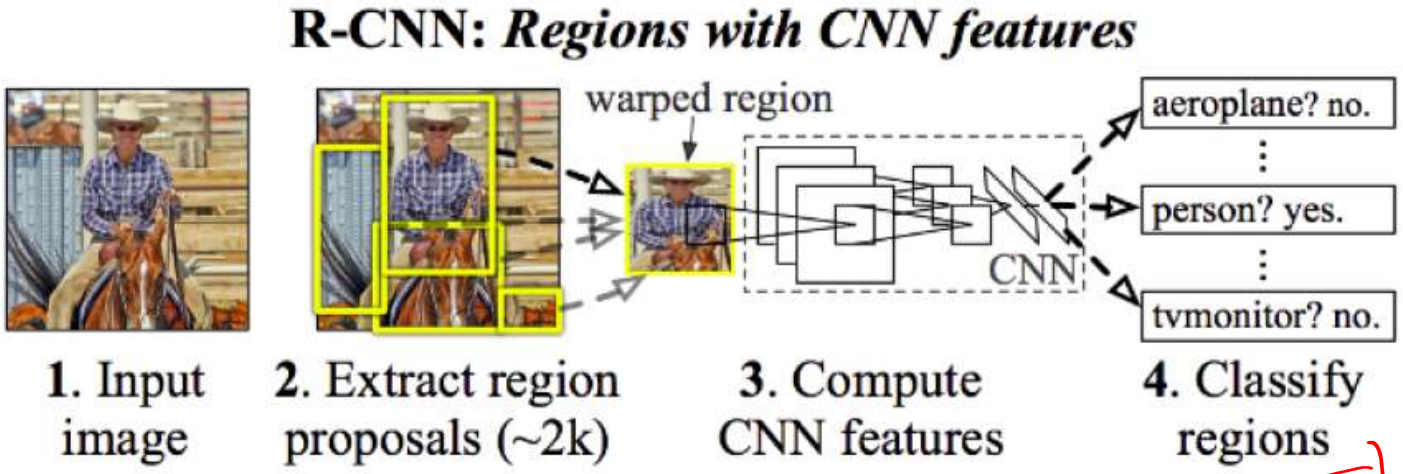
Cite 11049 Girshick, Ross and Donahue, Jeff and Darrell, Trevor and Malik, Jitendra, Rich feature hierarchies for accurate object detection and semantic segmentation, Conference on computer vision and pattern recognition, pages 580–587, IEEE-2014

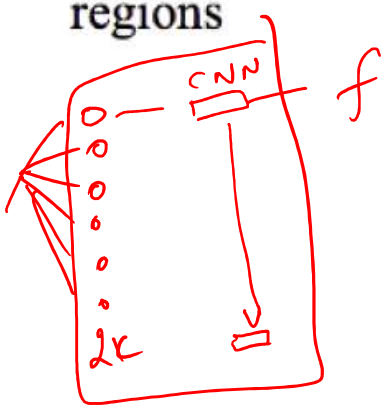
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CNN Case Studies (Fast R-CNN)

Dr. Kamlesh Tiwari

R-CNN

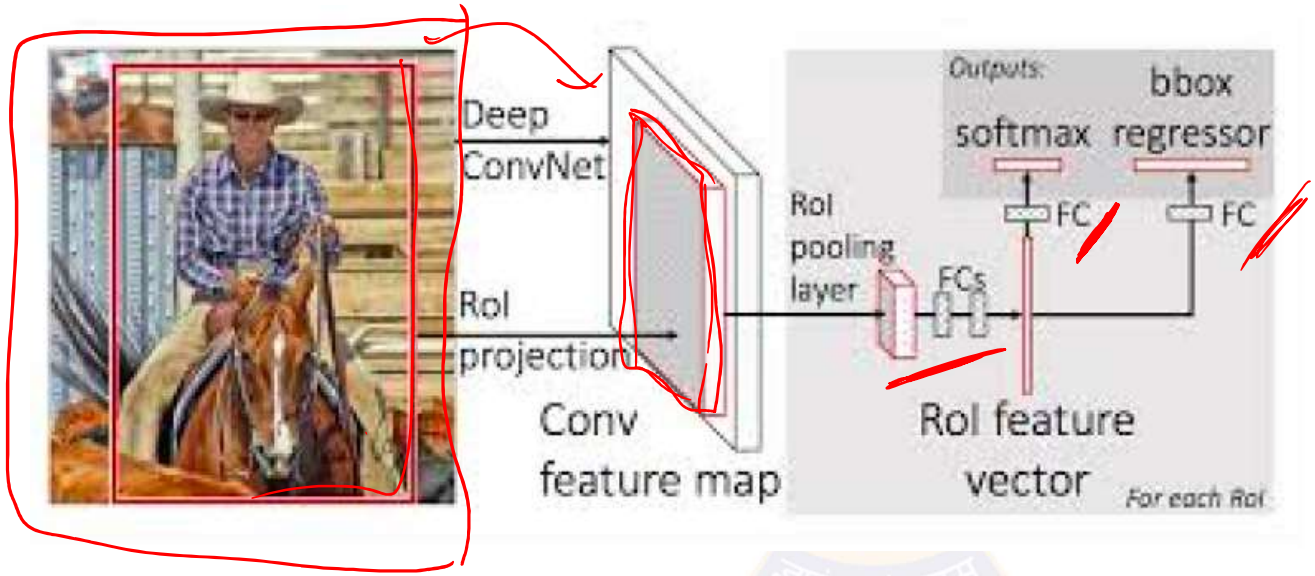




- Region proposals (from external selective search)
- Warped image (resizing)
- SVM for classification (one for each class)

Cite 11049 Girshick, Ross and Donahue, Jeff and Darrell, Trevor and Malik, Jitendra, Rich feature hierarchies for accurate object detection and semantic segmentation, Conference on computer vision and pattern recognition, pages 580–587, IEEE-2014

Fast R-CNN

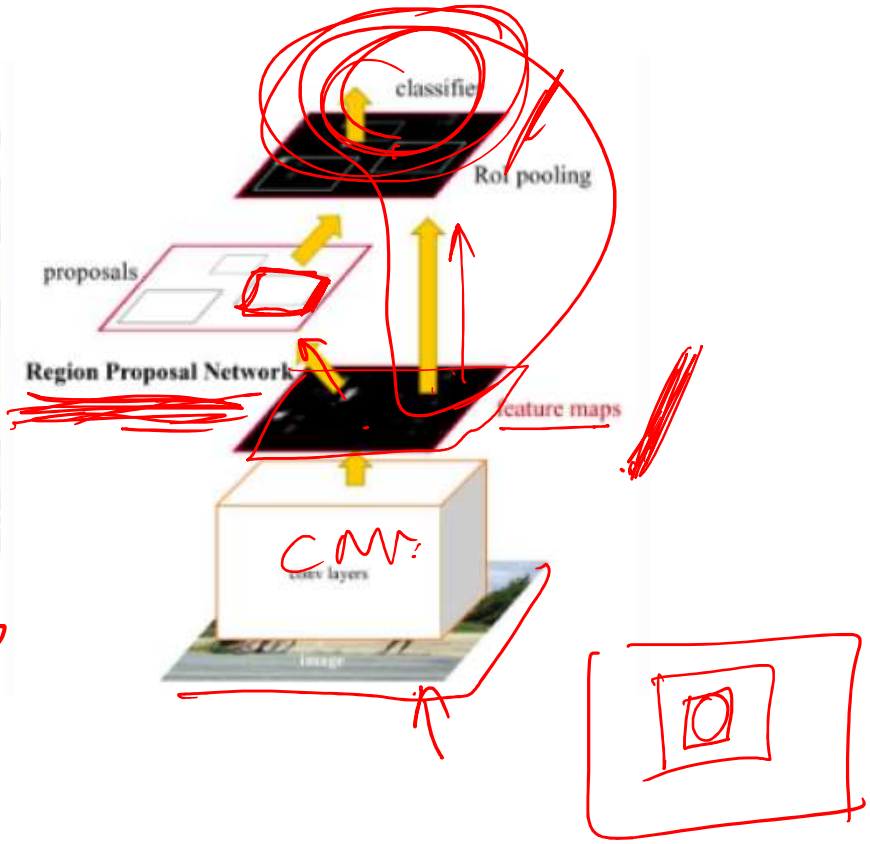


- RoI pooling
- Multi-task loss

Cite 7885 Girshick, Ross Fast R-CNN, International Conference on computer vision, pages 1440–1448, IEEE-2015

Faster R-CNN

- Region Proposal Network (RPN)
- Four loss: RPN classification loss, RPN regress loss, Final classification loss, Final box coordinate loss
- 250 time faster than R-CNN. (Fast R-CNN is 25 times fast)



Cite 7885 Girshick, Ross Fast R-CNN, International Conference on computer vision, pages 1440–1448, IEEE-2015

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

CNN Case Studies (Yolo)

Dr. Kamlesh Tiwari

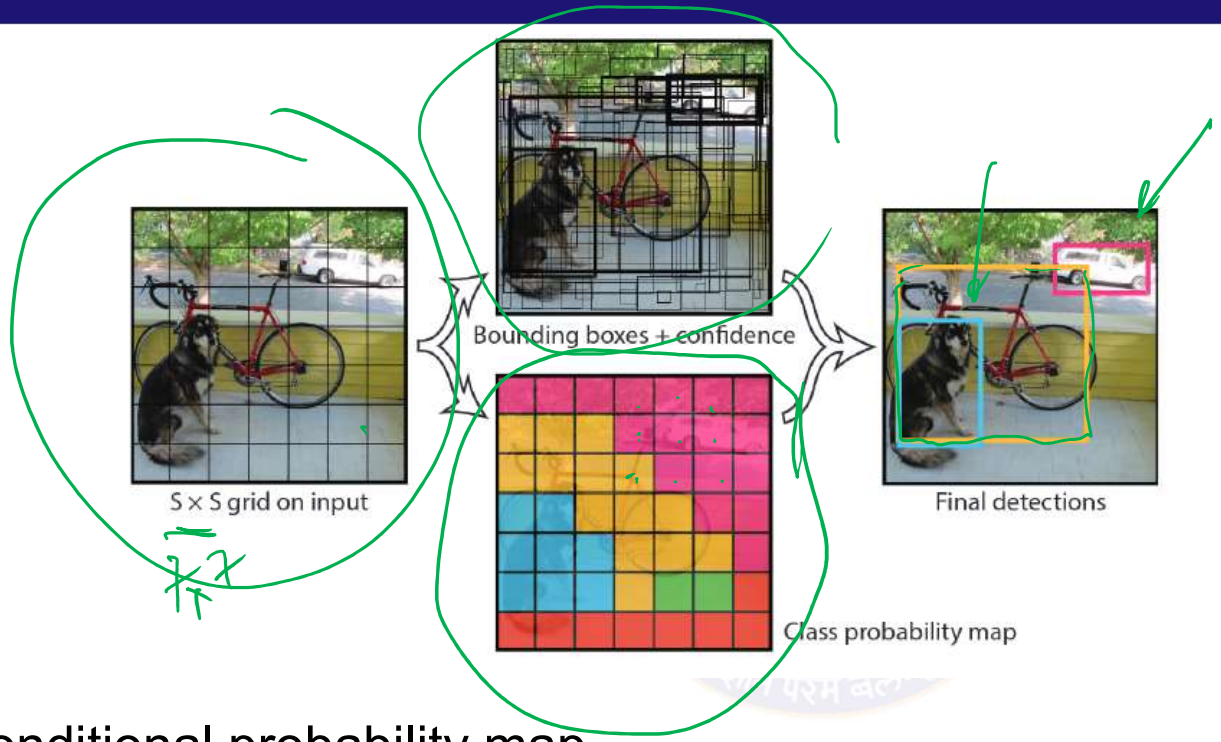
Object Detection with Yolo

- What is there and where?
- Deformative Part Model, and F-RCNN

- Apply the model to an image at multiple locations and scales.
- High scoring regions are considered detections.

Yolo: apply a single neural network to the full image that divides it into regions and predicts bounding boxes and probabilities for each region

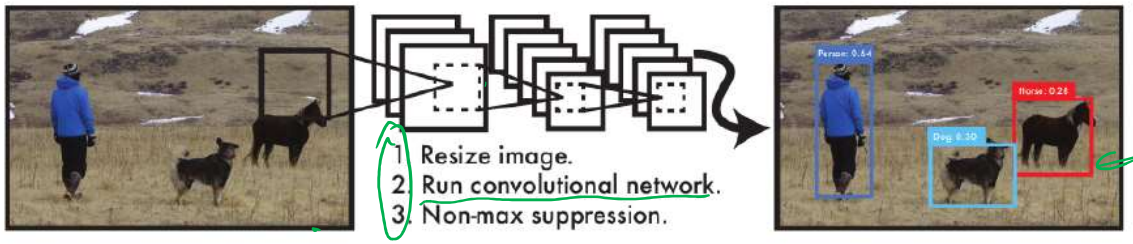
Yolo

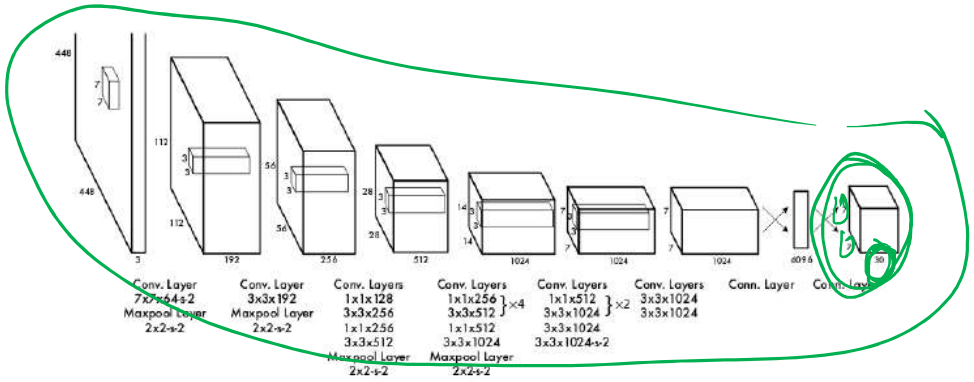


- Conditional probability map
- See https://pjreddie.com/darknet/yolo/

Cite 6375 Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali, You only look once: Unified, real-time object detection, IEEE conference on computer vision and pattern recognition (CVPR), pages 779–788, 2016

Yolo



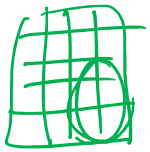


- SxS segments, gives B bounding boxes with confidence, and C class probabilities. So SxSx(Bx5 + C) values. S:7, B:2, C:20

Cite 6375 Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali, You only look once: Unified, real-time object detection, IEEE conference on computer vision and pattern recognition (CVPR), pages 779–788, 2016

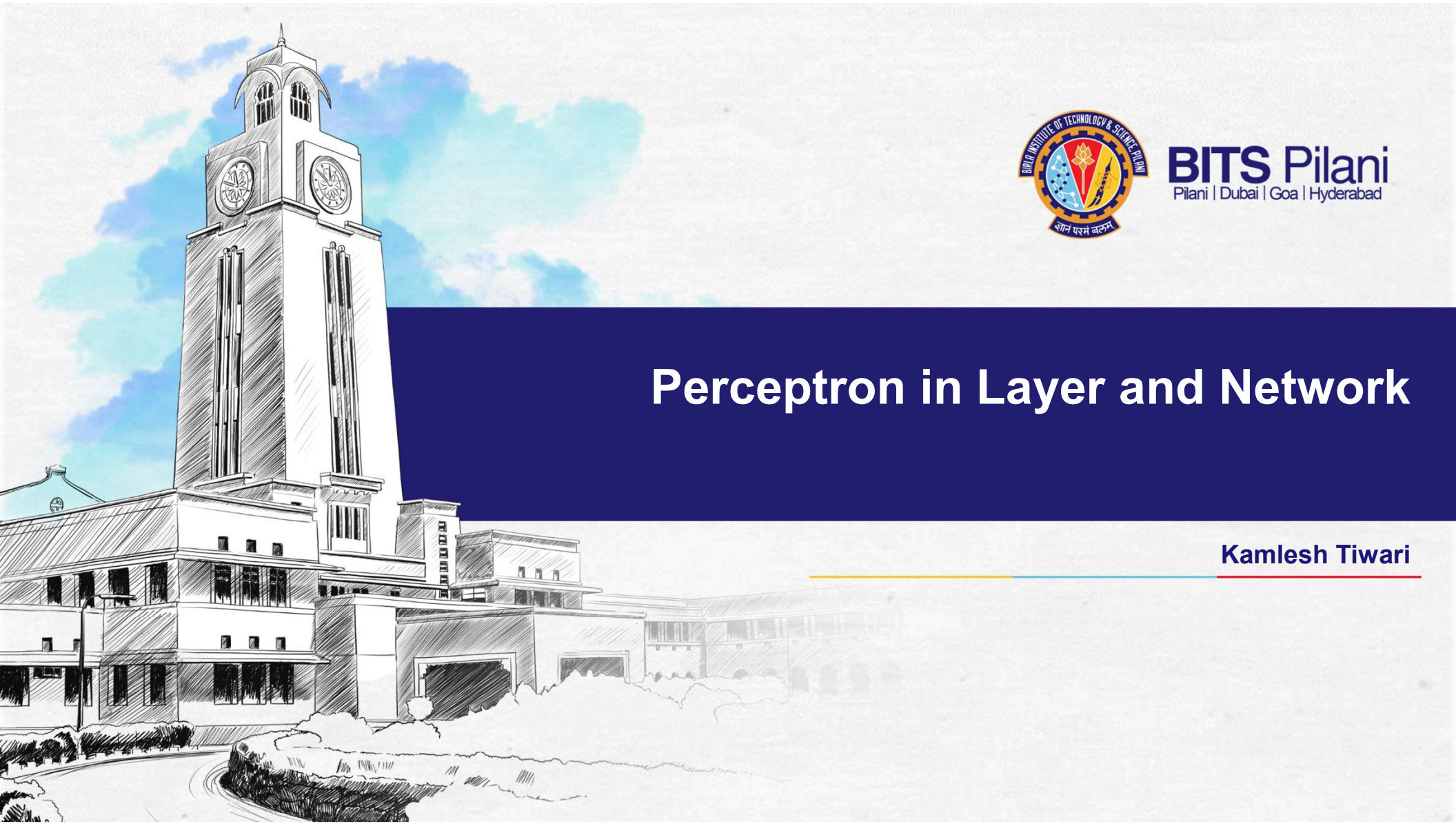
Yolo

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img



- It is fast
- Speed comes at the price of accuracy. Improved to 69%
- Generalizes well
- Latest version YOLOv3 2018

Cite 6375 Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali, You only look once: Unified, real-time object detection, IEEE conference on computer vision and pattern recognition (CVPR), pages 779–788, 2016

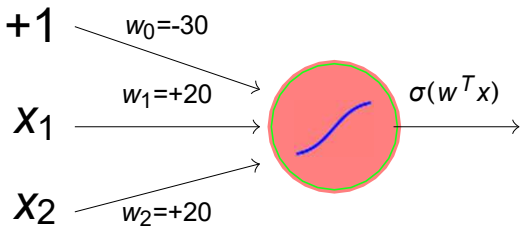


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Perceptron in Layer and Network

Kamlesh Tiwari

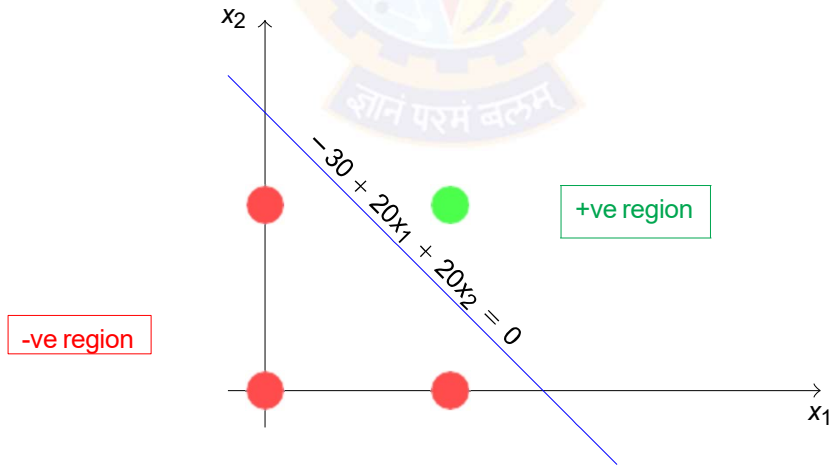
Essentially it Represents A Decision Boundary



Provides positive classification if

$$-30 + 20x_1 + 20x_2 \geq 0$$

Represents a linear decision boundary

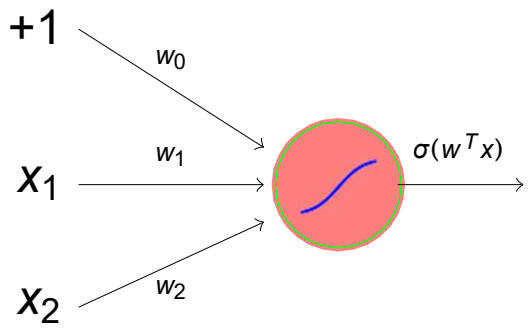


An Example

Design a perceptron for

X_1	x_2	Classification
0	0	0
0	1	0
1	0	1
1	1	0

Let us assume following



We have following four equations

$$w_0 + w_1 \times (0) + w_2 \times (0) < 0 \quad (1)$$
$$w_0 + w_1 \times (0) + w_2 \times (1) < 0 \quad (2)$$
$$w_0 + w_1 \times (1) + w_2 \times (0) \geq 0 \quad (3)$$
$$w_0 + w_1 \times (1) + w_2 \times (1) < 0 \quad (4)$$

By (1) $w_0 < 0$ so let $w_0 = -1$
By (2) $w_0+w_2<0$ so let $w_2 = -1$
By (3) $w_0+w_1 \geq 0$ so let $w_1 = 1.5$
By (4) $w_0+w_1+w_2<0$ that is valid

So $(w_0, w_1, w_2) = (-1, -1, 1.5)$

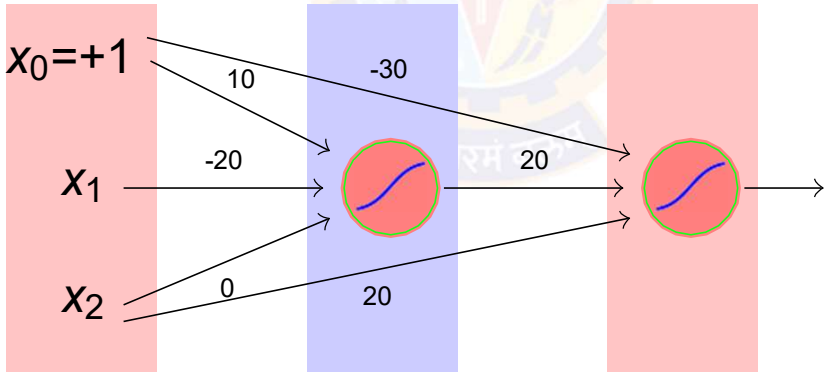
Other possibilities are also there

An Example

Design a neural network for

x_1	X_2	Classification
0	0	0
0	1	0
1	0	1
1	1	0

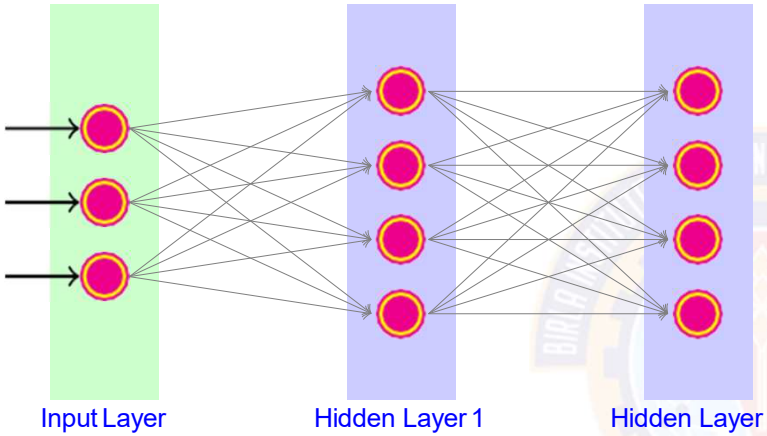
x_1	X_2	$\bar{x}_2$	$(x_1) AND (\bar{x}_2)$
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0



This arrangement is mostly avoided, as training is more challenging

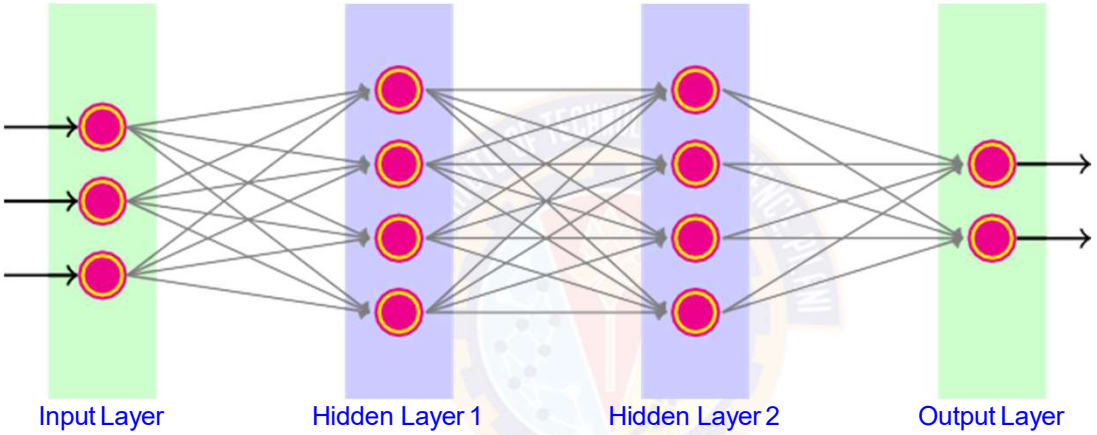
Neural Network

When neurons are interconnected in layers



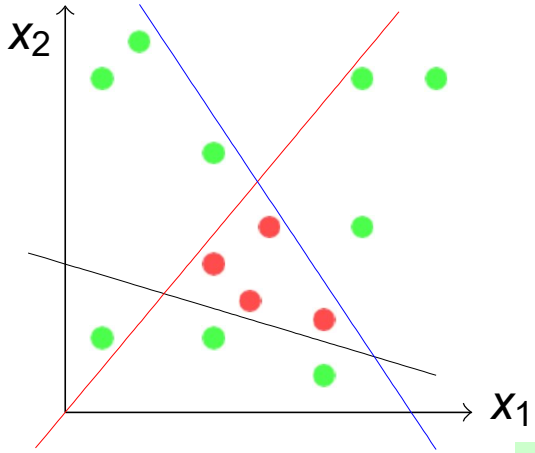
Neural Network

When neurons are interconnected in layers



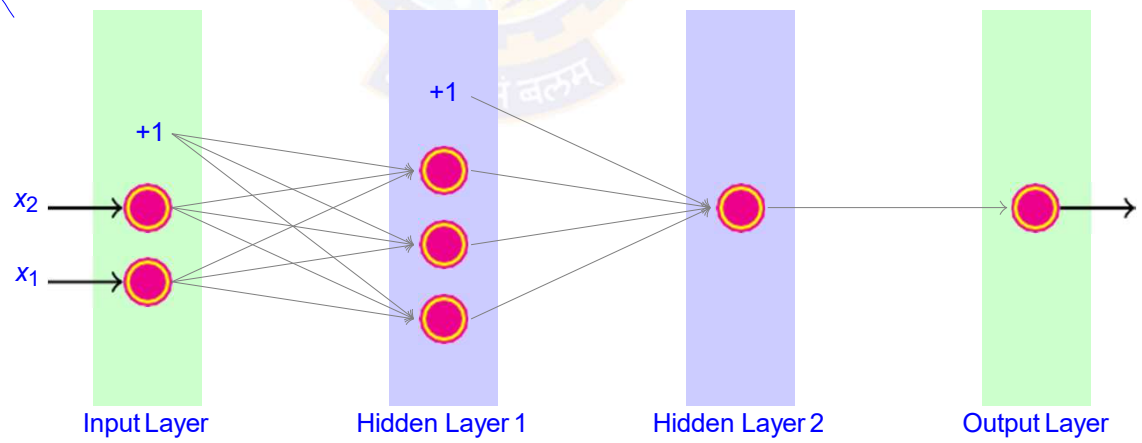
- Number of layers may differ
- Nodes in each intermediate layers may also differ
- Multiple output neurons are used for different class
- **Two levels deep** NN can represent any boolean function

More Example: Design NN for the following data



Whether it is green?

Red-line	Blue-line	Black-line	Color
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Neural Network Applications

- NN is appropriate for problems with the following characteristics:
  - Instances are provided by many attribute-value pairs (more data)
  - The target function output may be discrete-valued, real-valued, or a vector of several real or discrete valued attributes
  - The training examples may contain errors Long training times are acceptable
  - Fast evaluation of the target function may be required
  - The ability of humans to understand the learned target function is not important

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Perceptron Training (Delta Rule)

Kamlesh Tiwari

Perceptron Training (delta rule)

- Delta rule converges to a best-fit approximation of the target
- Uses gradient descent
- Consider unthresholded perceptron, $o(\vec{x}) = \vec{w}.\vec{x}$
- Training error is defined as

$$E(\vec{w}) = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$

- Gradient would specify direction of steepest increase

$$\nabla E(\vec{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$$

- Weights can be learned as $w_i = w_i - \eta \frac{\partial E}{\partial w_i}$
- It can be seen that $\frac{\partial E}{\partial w_i} = \sum_{d \in D}(t_d - o_d)(-x_{id})$

Perceptron Training (delta rule)

Algorithm 9: Gradient Descent (D,η)

```
1 Initialize w_i with random weights
2 repeat
3     For each w_i , initialize Δw_i = 0
4     for each training example d ∈D do
5         Compute output o using model for d whose target is t
6         For each w_i , update Δw_i = Δw_i + η(t−o)x_i
7     For each w_i , set w_i = w_i + Δw_i
8 until termination condition is met;
9 return w
```

Perceptron Training (delta rule)

Algorithm 10: Gradient Descent (D,η)

```
1 Initialize w_i with random weights
2 repeat
3     For each w_i , initialize Δw_i = 0
4     for each training example d ∈D do
5         Compute output o using model for d whose target is t
6         For each w_i , update Δw_i = Δw_i + η(t−o)x_i
7     For each w_i , set w_i = w_i + Δw_i
8 until termination condition is met;
9 return w
```

- A date item $d \in D$, is supposed to be multidimensional $d = (x_1, x_2, ..., x_n, t)$
- Algorithm converges toward the minimum error hypothesis.

Perceptron Training (delta rule)

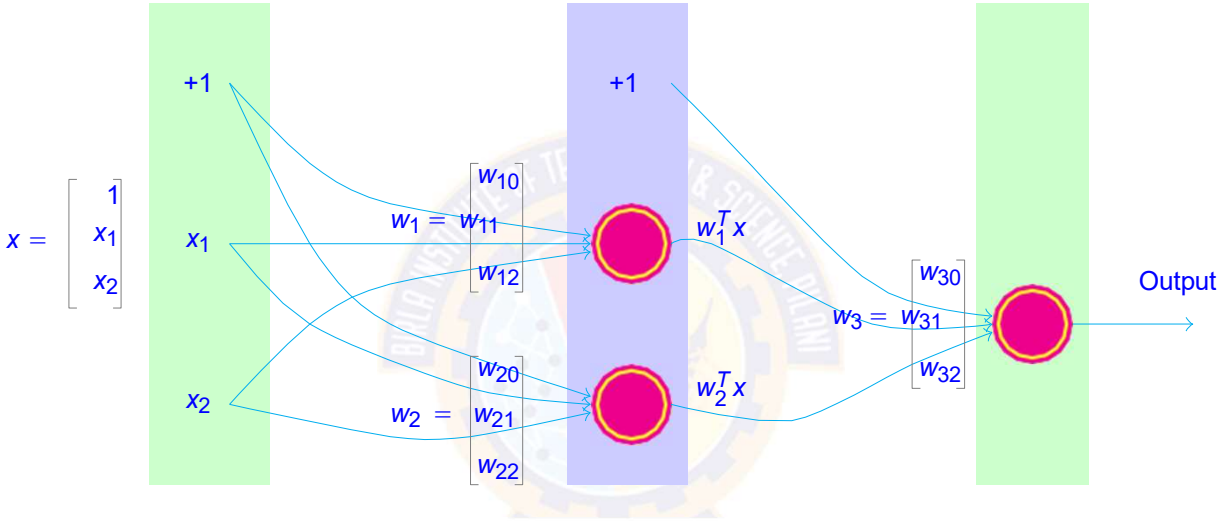
Algorithm 11: Gradient Descent (D,η)

```
1 Initialize w_i with random weights
2 repeat
3     For each w_i , initialize Δw_i = 0
4     for each training example d ∈D do
5         Compute output o using model for d whose target is t
6         For each w_i , update Δw_i = Δw_i + η(t−o)x_i
7     For each w_i , set w_i = w_i + Δw_i
8 until termination condition is met;
9 return w
```

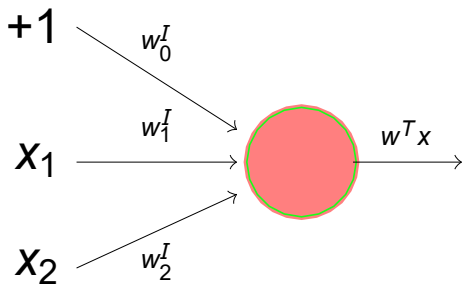
- A date item $d \in D$, is supposed to be multidimensional $d = (x_1, x_2, ..., x_n, t)$
- Algorithm converges toward the minimum error hypothesis.
- Linear programming can also be an an approach

Linear Activation is Not Much Interesting

NN with perceptrons have limited capability, even with many layers

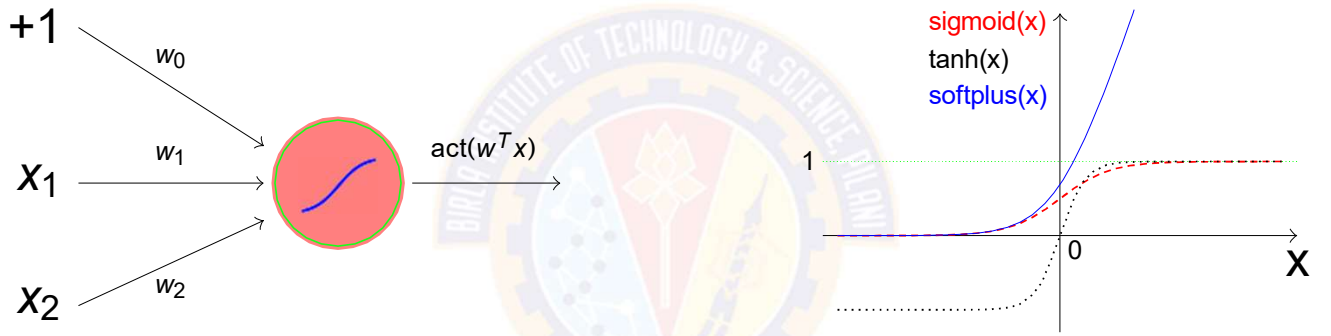


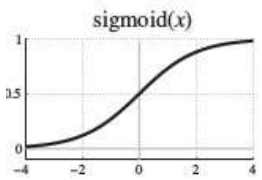
$$
\begin{aligned}
Output &= w_{30} \times 1 + w_{31} \times (w_1^T x) + w_{32} \times (w_2^T x) \\
&= w_{30} \times 1 + w_{31} \times [w_{10} \times 1 + w_{11} \times x_1 + w_{12} \times x_2] \\
&\quad + w_{32} \times [w_{20} \times 1 + w_{21} \times x_1 + w_{22} \times x_2] \\
&= (w_{30} + w_{31} w_{10} + w_{32} w_{20}) + (w_{31} w_{11} + w_{32} w_{21}) \times x_1 \\
&\quad + (w_{31} w_{12} + w_{32} w_{22}) \times x_2 \\
&= w_0' + w_1' \times x_1 + w_2' \times x_2
\end{aligned}
$$



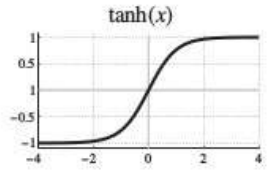
Expression of single perceptron

Neuron uses nonlinear **activation functions** (*sigmoid*, *tanh*, *ReLU*, *softplus etc.*) at the place of thresholding

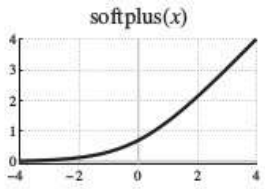




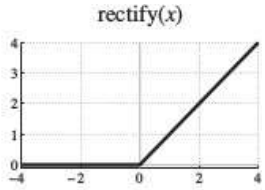
$$h(x) = \frac{1}{1 + \exp(-x)}$$



$$h(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



$$h(x) = \log(1 + \exp(x))$$



$$h(x) = \max(0, x)$$

Multilayer Networks and Backpropagation

- Single perceptron can only express linear decision surface
- We need units whose output is a nonlinear function of its inputs AND is also differentiable (Use Neuron not ~~Perceptron~~)

$$o(\vec{x}) = \sigma(\vec{w}.\vec{x})$$

where $\sigma(y) = \frac{1}{1+e^{-y}}$

- Backpropagation algorithm learns the weights for a fixed set of units and interconnections
- It employs gradient descent to minimize the error between the network output values and the target values for these outputs
- Let Error function is redefined as

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

Backpropagation (for 2 layers)

Algorithm 12: Backpropagation($D, \eta, n_{in}, n_{out}, n_{hidden}$)

1 Create the feedforward network with n_{in} , n_{out} , n_{hidden} layers
2 Randomly initialize weights to small values $\in [-0.05, +0.05]$
3 **repeat**
4 **for** *each* $<\vec{x}, \vec{t}> \in D$ **do**
5 $o_u =$ get output from network $\forall$ unit u
6 $\delta_k = o_k(1 - o_k)(t_k - o_k)$ for all output unit k
7 $\delta_h = o_h(1 - o_h) \sum_{k \in outputs} (w_{kh}\delta_k)$ for all hidden unit h
8 $w_{ji} = w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$
9 **until** *converge*;

- Recall error function is $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$

Backpropagation (for 2 layers)

Algorithm 13: Backpropagation($D, \eta, n_{in}, n_{out}, n_{hidden}$)

1 Create the feedforward network with n_{in} , n_{out} , n_{hidden} layers
2 Randomly initialize weights to small values $\in [-0.05, +0.05]$
3 **repeat**
4 **for** *each* $<\vec{x}, \vec{t}> \in D$ **do**
5 $o_u =$ get output from network $\forall$ unit u
6 $\delta_k = o_k(1 - o_k)(t_k - o_k)$ for all output unit k
7 $\delta_h = o_h(1 - o_h) \sum_{k \in outputs} (w_{kh}\delta_k)$ for all hidden unit h
8 $w_{ji} = w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$
9 **until** *converge*;

- Recall error function is $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$
- For a single training example $E_d(\vec{w}) = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$

Backpropagation (for 2 layers)

Algorithm 14: Backpropagation($D, \eta, n_{in}, n_{out}, n_{hidden}$)

1 Create the feedforward network with n_{in} , n_{out} , n_{hidden} layers
2 Randomly initialize weights to small values $\in [-0.05, +0.05]$
3 **repeat**
4 **for** *each* $<\vec{x}, \vec{t}> \in D$ **do**
5 $o_u =$ get output from network $\forall$ unit u
6 $\delta_k = o_k(1 - o_k)(t_k - o_k)$ for all output unit k
7 $\delta_h = o_h(1 - o_h) \sum_{k \in outputs} (w_{kh}\delta_k)$ for all hidden unit h
8 $w_{ji} = w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$
9 **until** *converge*;

- Recall error function is $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$
- For a single training example $E_d(\vec{w}) = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$

- Weight w_{ji} is updated by adding $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$

Thank You!

In our next session:

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Sequence Data

Kamlesh Tiwari

Sequence is Important

- Suppose you have a system, whom you can ask whether should you go to see a movie or NOT

  - The system need to consider many factors such as 1) who are the actors 2) what is the rating of the movie 3) your interests in type of movie 4) do you have money to purchase tickets *etc*.

  - Everyday it would give the same answer. How many times you can see the same movie?

Basic ML models consider only the current input (sometime it is useful to consider previous input/output as well)

Sequence Data is Everywhere

- There are various places we encounter Sequence Data

1. Sequence Prediction

2. Sequence Classification

3. Sequence Generation

4. Sequence-to-Sequence Prediction

Sequence Data is Everywhere

- There are various places we encounter Sequence Data

Sequence Prediction

Weather Forecasting
Stock Market Prediction
Product Recommendation

Sequence Classification

DNA Sequence Classification
Anomaly Detection
Sentiment Analysis

Sequence Generation

Text Generation
Music Generation
Image Captioning

Sequence-to-Sequence Prediction

Multi-Step Time Series Forecasting
Text Summarization
Language Translation

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Modeling Sequences

Kamlesh Tiwari

Sequences

Data is essentially a set of examples (every row represents one)

- Mostly we have fixed number of attributes in an example. However, some interesting applications (such as text, voice, video *etc*) have variable number of points
- These points could depend on each other in complicated way
- We want to do something like
  - Given a text we want to predict sentiment or next world
    Agra is a great place, one should go there and visit _____
  - Given a voice we want to recognize the speaker
  - Given a video we want to determine the activity Given a series (say stock values) predict next one

ISSUE: Basic ML models do **NOT** handle variable number of inputs

Modeling Sequences

There are some ideas to fix the issue (difficulty in using ML models)

- Consider only fixed length.
  (Unable to model long term dependencies)
- Bag of Words: use a vector of length equal to dictionary, and mark/count which words are present

  (since order is not preserved; following lines becomes same)

  David is good at math but is bad in science

  David is bad at math but is good in science

Modeling Sequences

There are some ideas to fix the issue (difficulty in using ML models)

- Consider only fixed length.
  (Unable to model long term dependencies)
- Bag of Words: use a vector of length equal to dictionary, and mark/count which words are present

  (since order is not preserved; following lines becomes same)

  David is good at math but is bad in science

  David is bad at math but is good in science

To model a sequence we need

- To deal with variable size input
- Maintain sequence order
- Keep track of long term dependencies
- Share parameters across the sequences

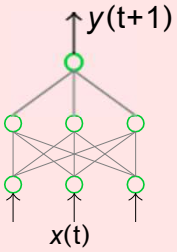
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Recurrent Neural Network (RNN)

Kamlesh Tiwari

Recurrent Networks (RNN)

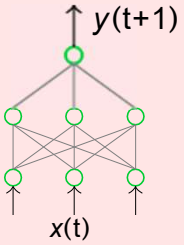
Feedforward network cannot capture the dependence of $y(t+1)$ on earlier values of x such as $x(t-1)$

Feedforward Network	Recurrent Network
	

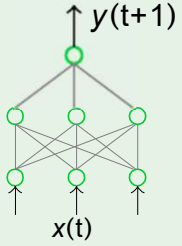
Recurrent Networks (RNN)

Feedforward network cannot capture the dependence of $y(t+1)$ on earlier values of x such as $x(t-1)$

Feedforward Network



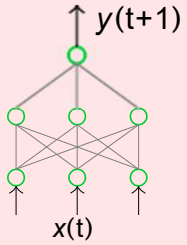
Recurrent Network



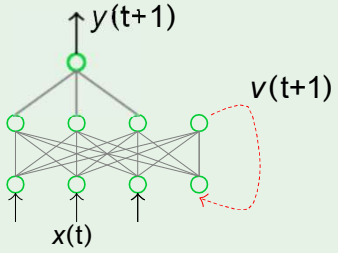
Recurrent Networks (RNN)

Feedforward network cannot capture the dependence of $y(t+1)$ on earlier values of x such as $x(t-1)$

Feedforward Network



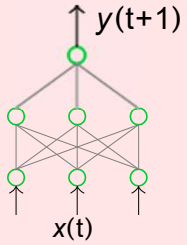
Recurrent Network



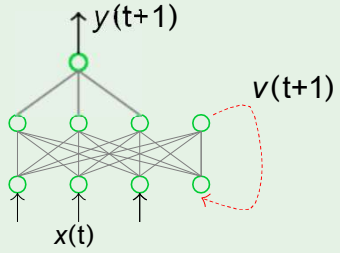
Recurrent Networks (RNN)

Feedforward network cannot capture the dependence of $y(t+1)$ on earlier values of x such as $x(t-1)$

Feedforward Network



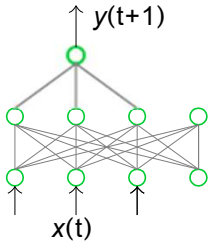
Recurrent Network



- RNN uses states (called self-state) of the network units available at time t as an input to the other units at time $t+1$
- RNN is suitable for temporal data (like time series)
- Training may involve unfolding and averaging.

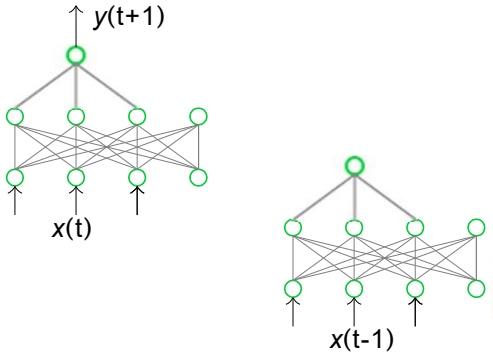
Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



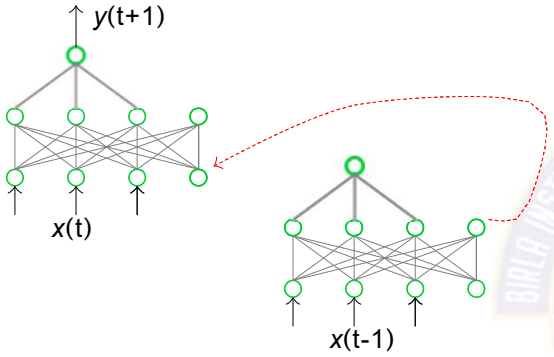
Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



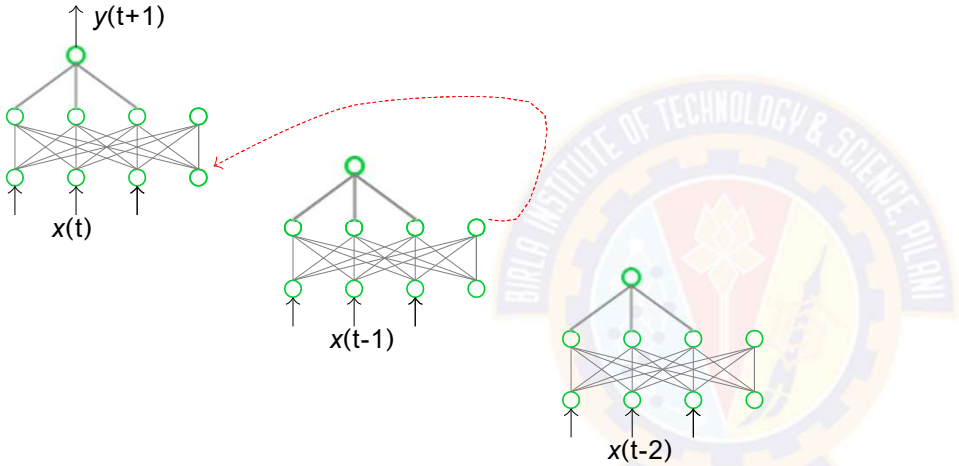
Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



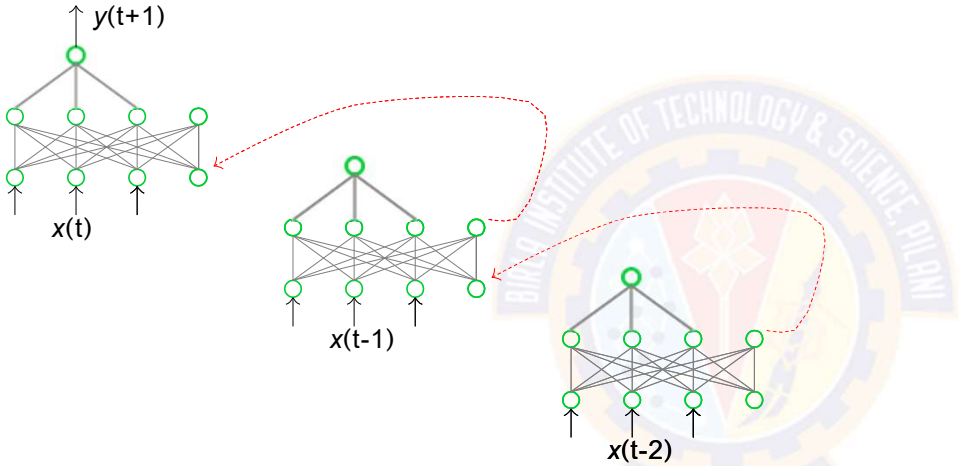
Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



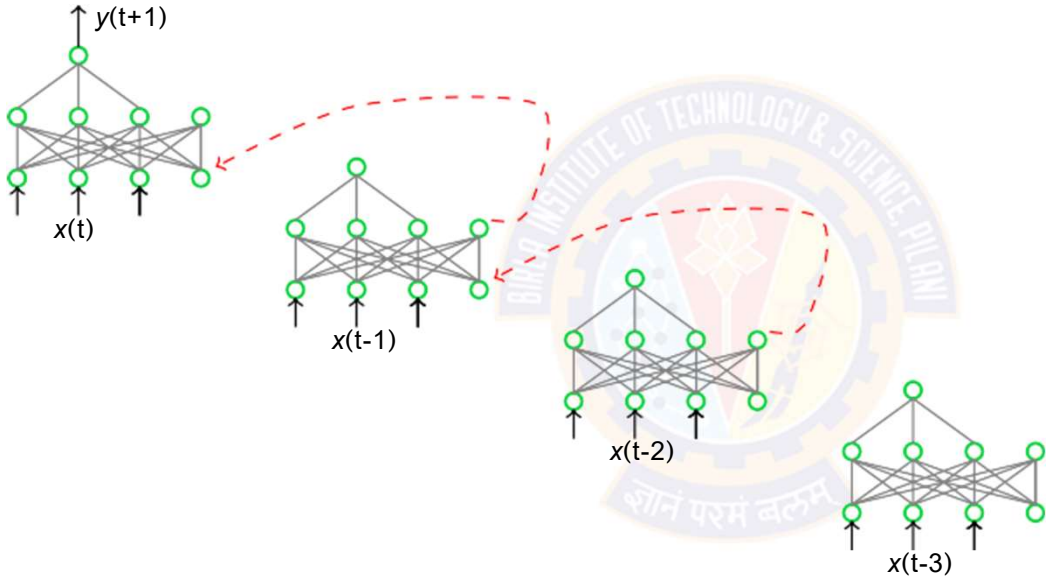
Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



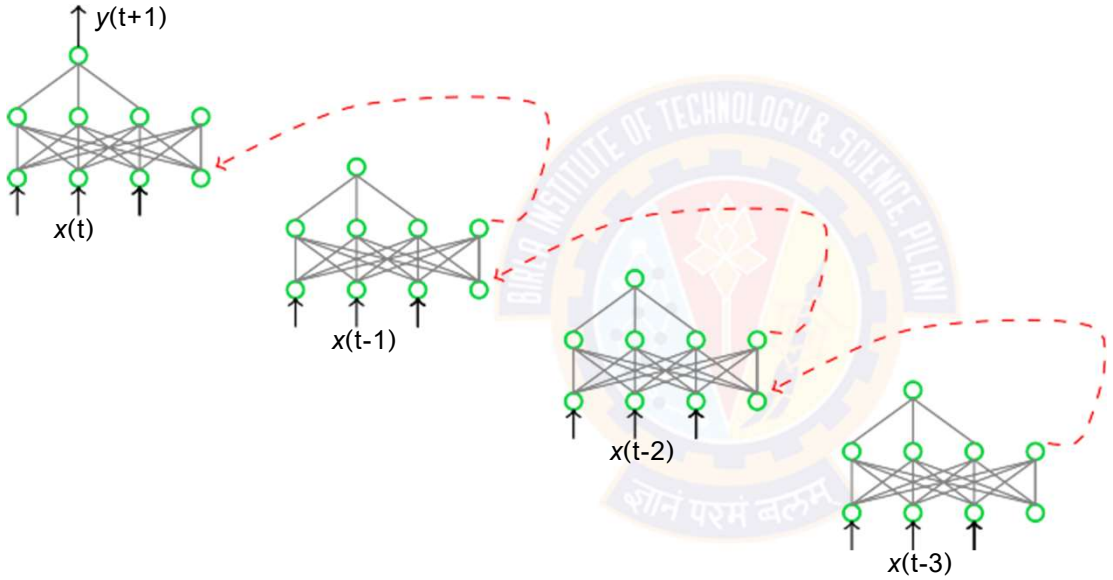
Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



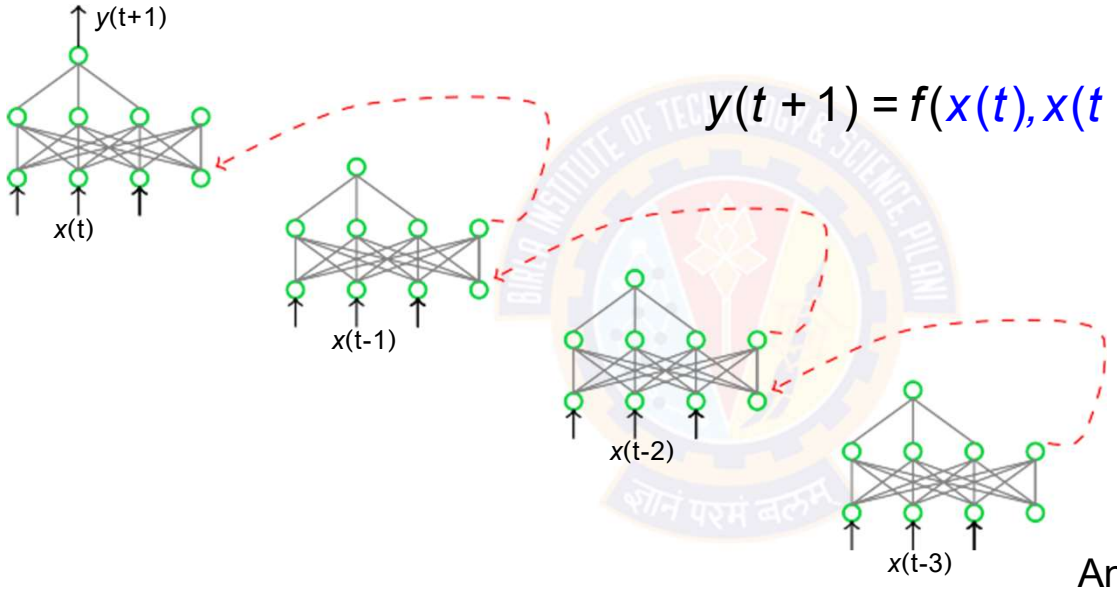
Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



Unfolding RNN in Time

When input at the time *t* is provided, what is output at time (*t* + 1) ?



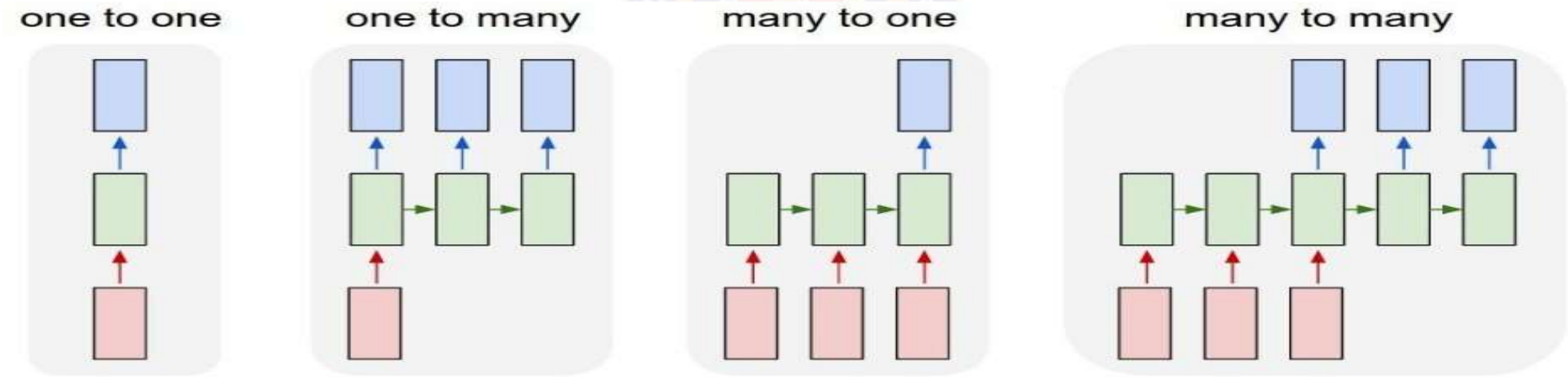
$$y(t+1) = f(x(t), x(t-1), \ldots, x(0))$$

And so on ...

By this way RNN incorporates history of the network in output

Various arrangements are possible based on need

- What should be the next world? **One-to-One**
- Caption the given image? **One-to-Many**
- Segmentation or classification**Many-to-One**
- Translate from one language to other**Many-to-Many**

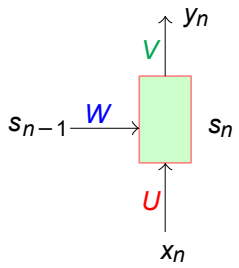


Output may be different for same input

We are optimizing over programs not on functions

Model and an application of RNN [1]

```
rnn = RNN()
y = rnn.step(x)
```

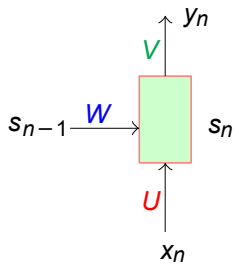



```
class RNN:
    def step (self, x):
        self.s = np.tanh(np.dot(self.W, self.s) + np.dot(self.U, x) y =
        np.dot(self.V, self.s)
        return y
```

[1] karpathy.github.io: Minimal character-level Vanilla RNN model

Model and an application of RNN [1]

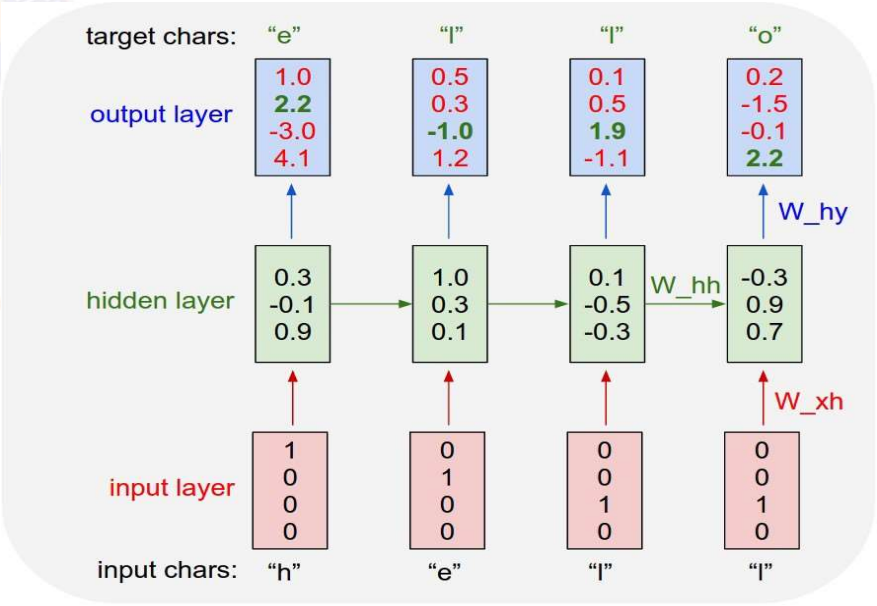
```
rnn = RNN()
y = rnn.step(x)
```




```
class RNN:
  def step (self, x):
      self.s = np.tanh(np.dot(self.W, self.s) + np.dot(self.U, x) y =
      np.dot(self.V, self.s)
      return y
```

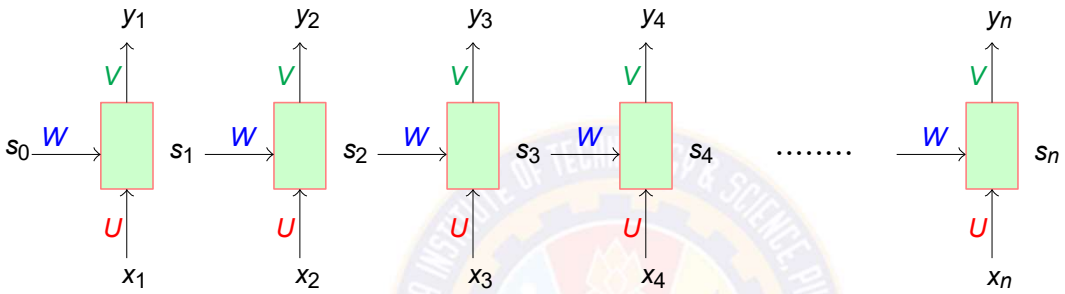
Character-Level Language Models

- Given a character what is the next character
- Characters are encoded using one-hot encoding
- Weights need to be adjusted



[1] karpathy.github.io: Minimal character-level Vanilla RNN model

Training: A Simplified Version of RNN



$$y_i = \sigma(V^T s_i) \qquad s_i = \sigma(U^T x_i + W^T s_{i-1})$$

If loss at time t be J_t

Then total loss is $J = \sum_t J_t$

J_t could be something like $-$ log(probability of true output)

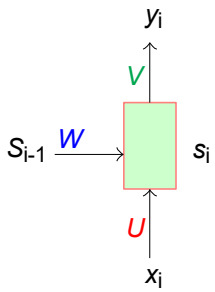
How to train RNN?

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Training RNN

Kamlesh Tiwari

Training RNN



$$y_i = \sigma(V^T s_i) \qquad s_i = \sigma(U^T x_i + W^T s_{i-1})$$

If loss at time t be J_t

Then total loss is $J = \sum_t J_t$

How to train RNN?

Training RNN

Backpropogation is used to train a RNN

- Total loss is the summation of J_t at every time step $J = \sum_t J_t$
- To train a parameter we need to find its gradient with respect to loss and shift the parameter in its opposite direction

$$W = W - \alpha \frac{\partial J}{\partial W} = W - \alpha \frac{\partial}{\partial W} \sum_t J_t = W - \alpha \sum_t \frac{\partial J_t}{\partial W}$$

- Consider a single summation term

$$\frac{\partial J_i}{\partial W} = \frac{\partial J_i}{\partial s_i} \times \frac{\partial s_i}{\partial W}$$

- Term $\frac{\partial s_i}{\partial W}$ is complicated. It depends on $s_{i-1}, s_{i-2}, \ldots, s_1$

Training RNN

- As $s_i = \sigma(U^T x_i + W^T s_{i-1})$
- where s_i also depends on W and other previous states $s_{i-2}, \ldots, s_1$
- Its derivative have **explicit** and **implicit** parts. Explicit part (we represent as ∂^+) considers other things as constant

Training RNN

- As $s_i = \sigma(U^T x_i + W^T s_{i-1})$
- where s_i also depends on W and other previous states $s_{i-2}, \ldots, s_1$
- Its derivative have **explicit** and **implicit** parts. Explicit part (we represent as ∂^+) considers other things as constant

$$
\begin{aligned}
\frac{\partial s_i}{\partial W} &= \frac{\partial^+ s_i}{\partial W} + \frac{\partial s_i}{\partial s_{i-1}} \frac{\partial s_{i-1}}{\partial W} \\
&= \frac{\partial^+ s_i}{\partial W} + \frac{\partial s_i}{\partial s_{i-1}} \left[\frac{\partial^+ s_{i-1}}{\partial W} + \frac{\partial s_{i-1}}{\partial s_{i-2}} \frac{\partial s_{i-2}}{\partial W} \right] \\
&= \frac{\partial^+ s_i}{\partial W} + \frac{\partial s_i}{\partial s_{i-1}} \frac{\partial^+ s_{i-1}}{\partial W} + \frac{\partial s_i}{\partial s_{i-1}} \frac{\partial s_{i-1}}{\partial s_{i-2}} \left[\frac{\partial^+ s_{i-2}}{\partial W} + \frac{\partial s_{i-2}}{\partial s_{i-3}} \frac{\partial s_{i-3}}{\partial W} \right] \\
&= \sum_{k=1}^{i} \frac{\partial s_i}{\partial s_k} \frac{\partial^+ s_k}{\partial W}
\end{aligned}
$$

where we use $\frac{\partial s_i}{\partial s_k}$ as a short form for $\frac{\partial s_i}{\partial s_{i-1}} \frac{\partial s_{i-1}}{\partial s_{i-2}} \frac{\partial s_{i-2}}{\partial s_{i-3}} \cdots \frac{\partial s_{k+2}}{\partial s_{k+1}} \frac{\partial s_{k+1}}{\partial s_k}$

Exploding and Vanishing Gradient

Consider $\frac{\partial s_i}{\partial s_k}$ that is $\frac{\partial s_i}{\partial s_{i-1}} \frac{\partial s_{i-1}}{\partial s_{i-2}} \frac{\partial s_{i-2}}{\partial s_{i-3}} \ldots \frac{\partial s_{k+2}}{\partial s_{k+1}} \frac{\partial s_{k+1}}{\partial s_k} = \prod_{j=i}^{k+1} \frac{\partial s_j}{\partial s_{j-1}}$

Focus on a single term $\frac{\partial s_j}{\partial s_{j-1}}$

it can be shown that it is upper-bounded by some constant (which depends on W)
let's call that value c

Since $\frac{\partial s_i}{\partial s_k}$ is a multiplication of $i - k$ such terms (and we expect $i - k$ to be large to address long term dependency) therefore the value

$$\frac{\partial s_i}{\partial s_k} \rightarrow \begin{cases} \text{Vanish} & \text{if } c < 1 \\ \text{Explode} & \text{if } c > 1 \end{cases}$$

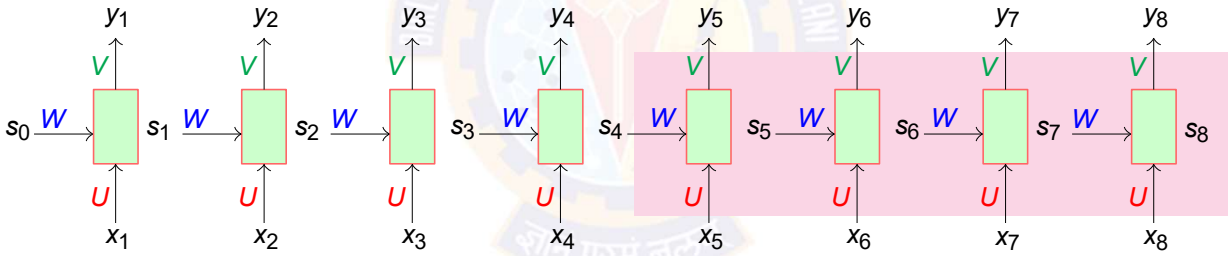
$(0.90 \rightarrow 0.81 \rightarrow 0.73 \rightarrow 0.66 \rightarrow 0.59 \rightarrow 0.53 \rightarrow 0.48 \rightarrow 0.43 \rightarrow 0.39)$
$(1.50 \rightarrow 2.25 \rightarrow 3.38 \rightarrow 5.06 \rightarrow 7.59 \rightarrow 11.39 \rightarrow 17.09 \rightarrow 25.63)$

Avoiding Exploding and Vanishing Gradient

- **Clipping**

  Try normalizing value of c keeping it some range $T_l \leq c \leq Th$

- **Truncated Backpropogation**
  At any step look for only for last k timestamps



- **LSTM**
  Special circuits for handling long term dependencies

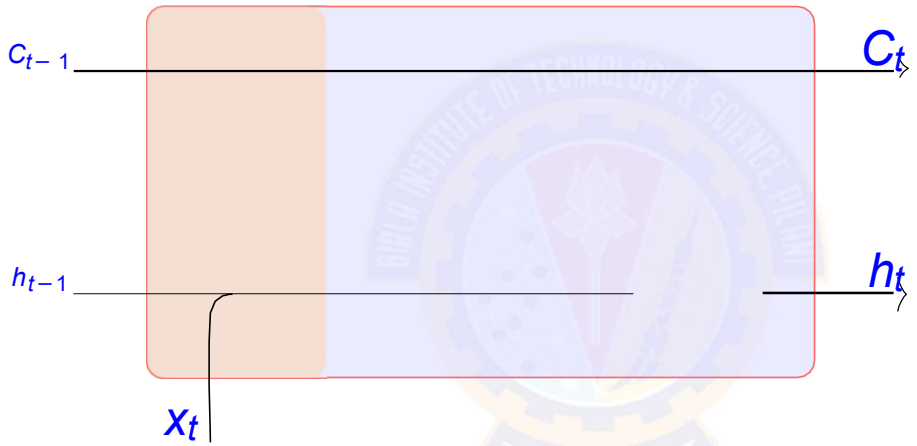
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

LSTM

Kamlesh Tiwari

LSTM: Long Short Term Memory [2]

LSTM are RNN with 4 special NN circuits for forget, store and output



[2] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

LSTM: Long Short Term Memory [2]

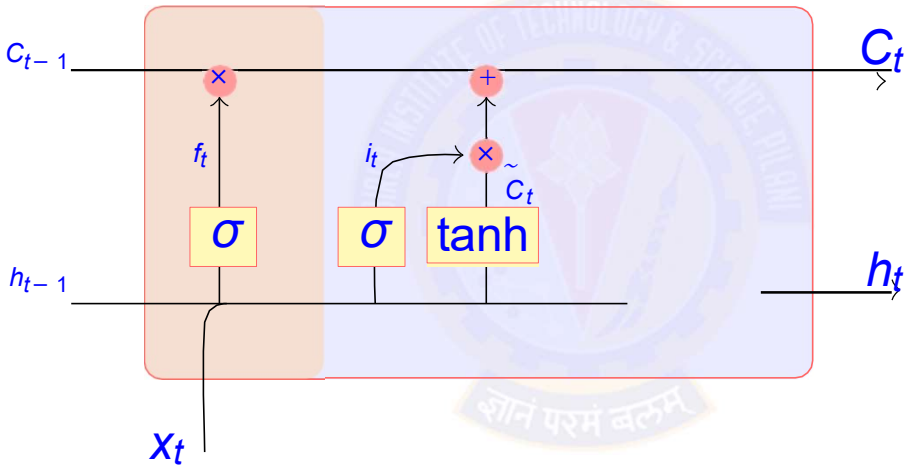
LSTM are RNN with 4 special NN circuits for forget, store and output

$$f_t = \sigma(W_f[h_{t-1}x_t] + b_f)$$

[2] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

LSTM: Long Short Term Memory [2]

LSTM are RNN with 4 special NN circuits for forget, store and output



$$f_t = \sigma(W_f[h_{t-1}x_t] + b_f)$$

$$i_t = \sigma(W_i[h_{t-1}x_t] + b_i)$$

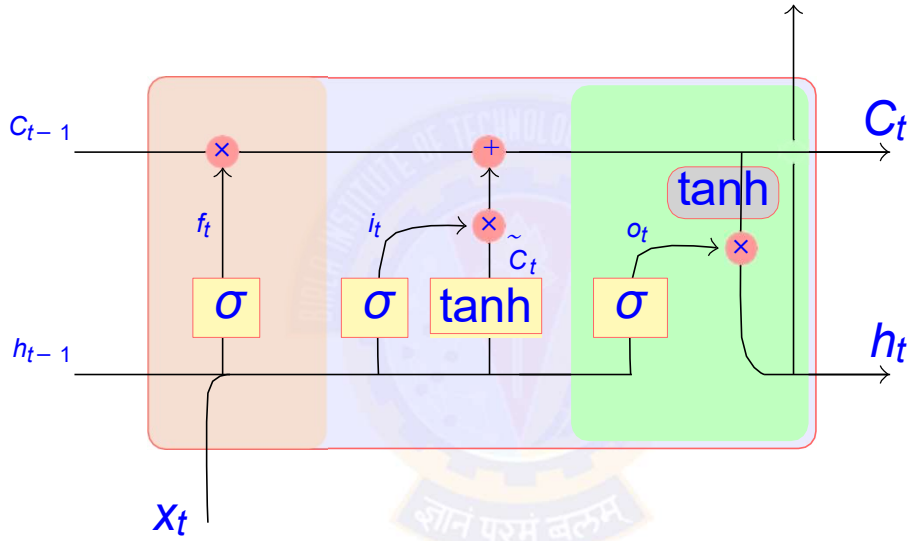
$$\tilde{C}_t = \tanh(W_c[h_{t-1}x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

[2] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

LSTM: Long Short Term Memory [2]

LSTM are RNN with 4 special NN circuits for forget, store and output



$$f_t = \sigma(W_f[h_{t-1}x_t] + b_f)$$

$$i_t = \sigma(W_i[h_{t-1}x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o[h_{t-1}x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

[2] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

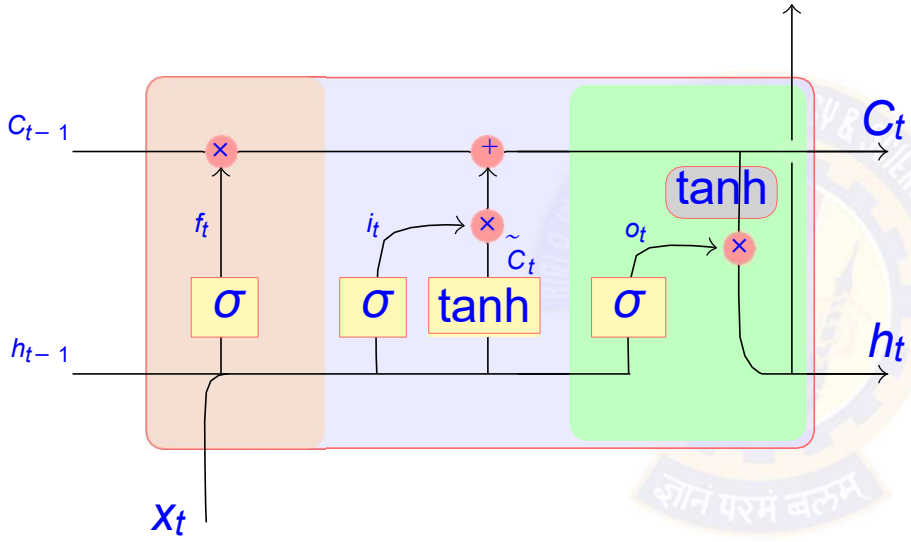
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Applications of LSTM

Kamlesh Tiwari

LSTM: Long Short Term Memory

LSTM are RNN with 4 special NN circuits for forget, store and output



$$f_t = \sigma(W_f[h_{t-1}x_t] + b_f)$$

$$i_t = \sigma(W_i[h_{t-1}x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o[h_{t-1}x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Hochreiter, Sepp, and Jrgen Schmidhuber. ”Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.

Example-01: what is at a specific place?

Five numbers are fed and we want the 2nd one as output.

```
from random import randint
from numpy import array
from numpy import argmax
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense

def generate_sequence(length, n_features):
    return [randint(0, n_features-1) for _ in range(length)]

def one_hot_encode(sequence, n_features):
    encoding = list()
    for value in sequence:
        vector = [0 for _ in range(n_features)]
        vector[value] = 1
        encoding.append(vector)
    return array(encoding)

def one_hot_decode(encoded_seq):
    return [argmax(vector) for vector in encoded_seq]

def generate_example(length, n_features, out_index):
    sequence = generate_sequence(length, n_features)
    encoded = one_hot_encode(sequence, n_features)
    X = encoded.reshape((1, length, n_features))
    y = encoded[out_index].reshape(1, n_features)
    return X, y
```

```
### Defining the model ########
length = 5
n_features = 10
out_index = 2
model = Sequential()
model.add(LSTM(25, input_shape=(length, n_features)))
model.add(Dense(n_features, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
model.summary()

#####  Fitting the model ############
for i in range(10000):
    X, y = generate_example(length, n_features, out_index)
    model.fit(X, y, epochs=1, verbose=2)

#####  Evaluating the model  ############
correct = 0
for i in range(100):
    X, y = generate_example(length, n_features, out_index1)
    yhat = model.predict(X)
    if one_hot_decode(yhat) == one_hot_decode(y1):
        correct += 1
print('Accuracy: %f' % ((correct/100)*100.0))
```

The system have achieved 100% accuracy

Example-02: Shakespeare like text

- All works of Shakespeare (a single 4.4MB file) fed to a 3-layer RNN with 512 hidden nodes on each layer. Output is as below

- Original

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

- RNN output

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

Example-02

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.utils import np_utils

## Data Preprocessing ###
data = open("a.txt").read().lower()
chars = sorted(list(set(data)))
print(chars)
totalChars = len(data)
print(totalChars)
numberOfUniqueChars = len(chars)
print(numberOfUniqueChars)
CharsForids = {char:Id for Id, char in enumerate(chars)}
idsForChars = {Id:char for Id, char in enumerate(chars)}
numberOfCharsToLearn = 100

#### Preprocessing #######
charX = []
y = []
counter = totalChars - numberOfCharsToLearn
for i in range(0, counter, 1):
    theInputChars = data[i:i+numberOfCharsToLearn]
    theOutputChars = data[i + numberOfCharsToLearn]
    charX.append([CharsForids[char] for char in theInputChars])
    y.append(CharsForids[theOutputChars])
X = np.reshape(charX, (len(charX), numberOfCharsToLearn, 1))
X = X/float(numberOfUniqueChars)
y = np_utils.to_categorical(y)
```

```
## Building the RNN model ####
model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax')) #number of features on the output
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.fit(X, y, epochs=500, batch_size=128)
model.save_weights("Othello.hdf5")

## Generating New Text ###
randomVal = np.random.randint(0, len(charX)-1)
randomStart = charX[randomVal]
print(randomStart)
for i in range(100):
        x = np.reshape(randomStart, (1, len(randomStart), 1))
        x = x/float(numberOfUniqueChars)
        pred = model.predict(x)
        index = np.argmax(pred)
        randomStart.append(index)
        randomStart = randomStart[1: len(randomStart)]
print("".join([idsForChars[value] for value in randomStart]))
```

```
## Sample contents from "a.txt" ###
Still questioned me the story of my life
From year to year – the battles, sieges, fortunes
That I have passed.
I ran it through, even from my boyish days
To th' very moment that he bade me tell it.
Wherein I spoke of most disastrous chances,
Of moving accidents by flood and field;
Of hairbreadth scapes i' the' imminent deadly breach;
```

https://github.com/sagar448/Keras-Recurrent-Neural-Network-Python/blob/master/README.md

Example-03: Algebraic geometric tex file

- Training a multilayer LSTM on *the stacks project* [5] online Latex code (16MB) for Algebraic Geometry book produced

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciely in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) =$ $\mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation ??. Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

[5] https://stacks.math.columbia.edu/

Example-04: Training with Linux Source Code [6]

- 474MB C source code in the *Linux repo on Github* trained on 3-layer LSTMs produced

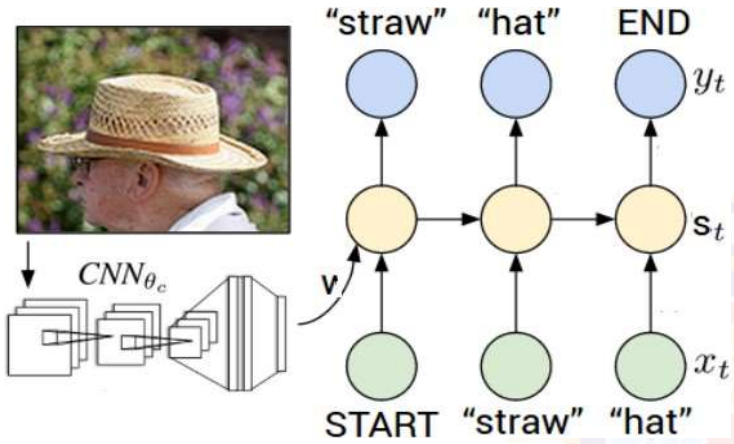
```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

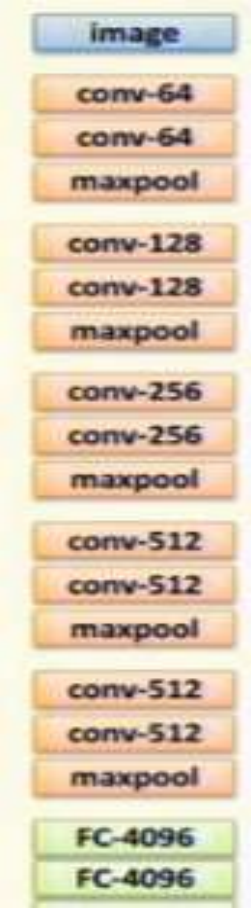
```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  div_u64_w(val, inb_p);
  spin_unlock(&disk->queue_lock);
  mutex_unlock(&s->sock->mutex);
  mutex_unlock(&func->mutex);
  return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
  pci_disable_spool(port);
  return 0;
}

static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

Example-05: Image Captioning [7]





Before:

$$s_t = \tanh(U^T x_i + W^T s_{i-1})$$

Now:

$$s_t = \tanh(U^T x_i + W^T s_{i-1} + Z^T v)$$

[7]Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

Thank You!

In our next session:

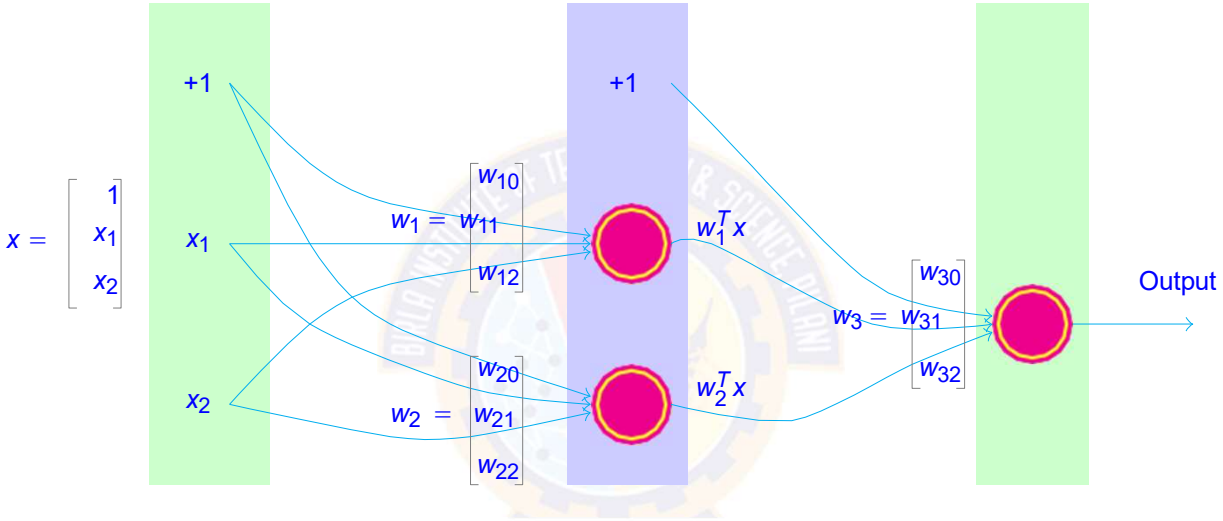
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Importance of Non-Linearity

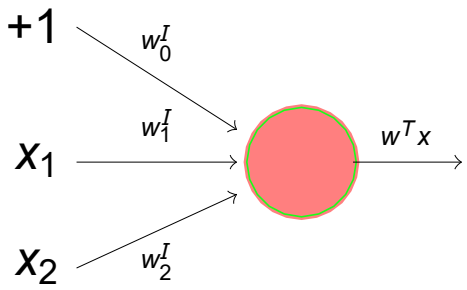
Dr. Kamlesh Tiwari

Linear Activation is Not Much Interesting

NN with perceptrons have limited capability, even with many layers



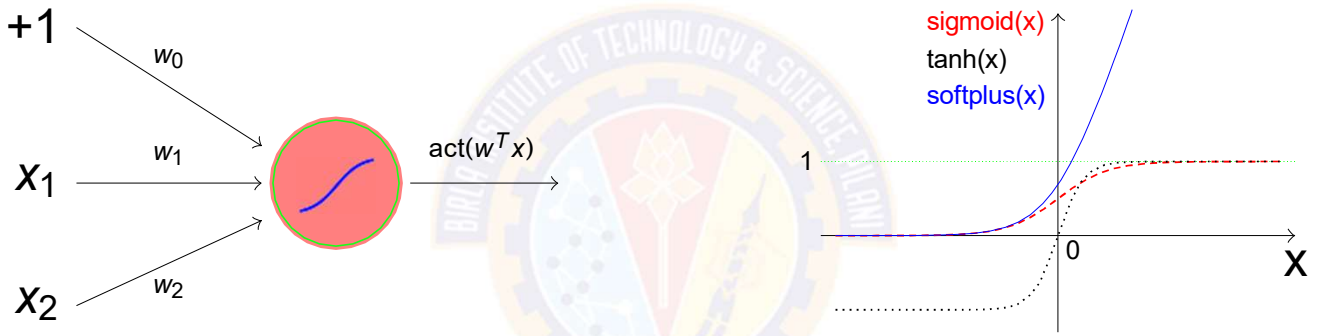
$$
\begin{aligned}
Output &= w_{30} \times 1 + w_{31} \times (w_1^T x) + w_{32} \times (w_2^T x) \\
&= w_{30} \times 1 + w_{31} \times [w_{10} \times 1 + w_{11} \times x_1 + w_{12} \times x_2] \\
&\quad + w_{32} \times [w_{20} \times 1 + w_{21} \times x_1 + w_{22} \times x_2] \\
&= (w_{30} + w_{31} w_{10} + w_{32} w_{20}) + (w_{31} w_{11} + w_{32} w_{21}) \times x_1 \\
&\quad + (w_{31} w_{12} + w_{32} w_{22}) \times x_2 \\
&= w_0' + w_1' \times x_1 + w_2' \times x_2
\end{aligned}
$$

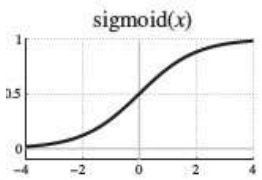


Expression of single perceptron

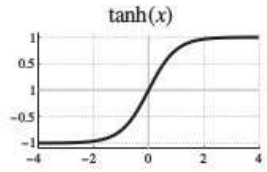
Neuron

Neuron uses nonlinear activation functions (*sigmoid*, *tanh*, *ReLU*, *softplus etc.*) at the place of thresholding

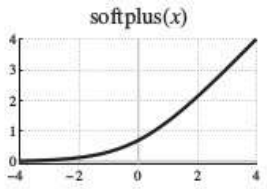




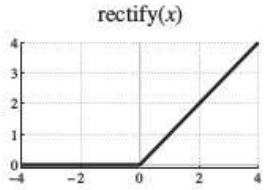
$$h(x) = \frac{1}{1 + \exp(-x)}$$



$$h(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



$$h(x) = \log(1 + \exp(x))$$



$$h(x) = \max(0, x)$$

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

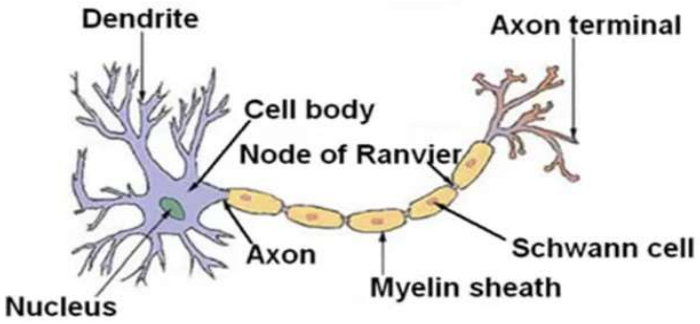
Introduction to Perceptron

Kamlesh Tiwari

Neural Network (NN)

NN is biologically motivated learning model that mimic human brain

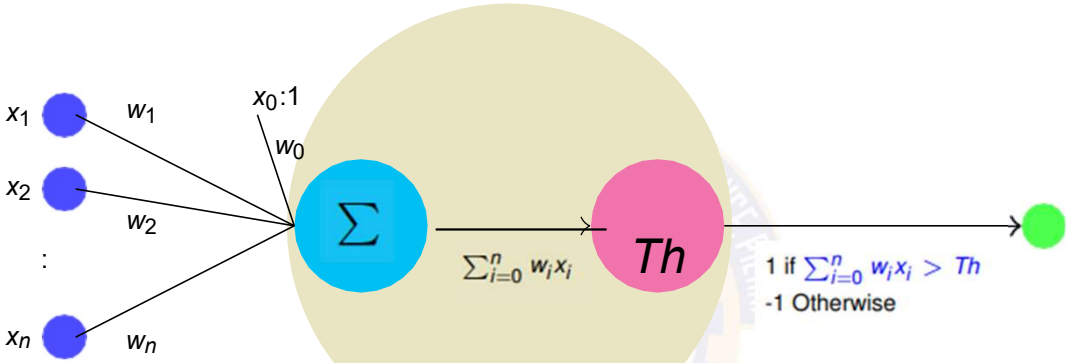
- Started by *W. McCulloch* study on working of neurons in 1943
- MADALINE (1959), an adaptive filter that eliminates echoes on phone lines was the first neural network
- Popularity of Neural Network diminished in 90's but, due to advances in **processing power** and availability of **large data** it again became state-of-the-art



- Cell, Axon, Synopses, Molicules, and Dendrites
- Humans have 10^{11} neurons, each connected to 10^{4} others, switches in 10^{-3} sec

A Single Perceptron

Perceptron representation

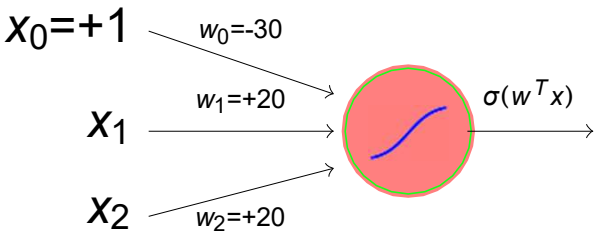


- A single perceptron can represent many boolean functions
- Any *m*-of-*n* function (at least *m* of the *n* inputs must be true) can be represented by perceptron. OR (*m*=1) and AND (*m*=*n*)

Two layer NN can represent any boolean function (Consider SOP)

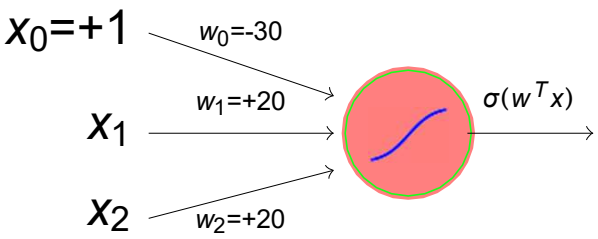
An Example

Consider a perceptron with output 0/1 as below



An Example

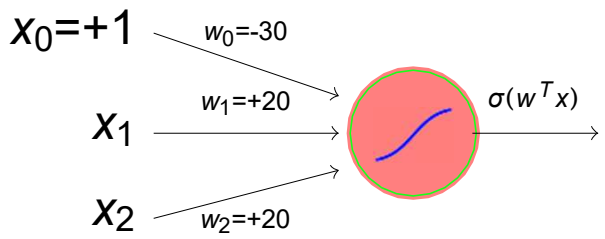
Consider a perceptron with output 0/1 as below



x_1	x_2	Output
0	0	

An Example

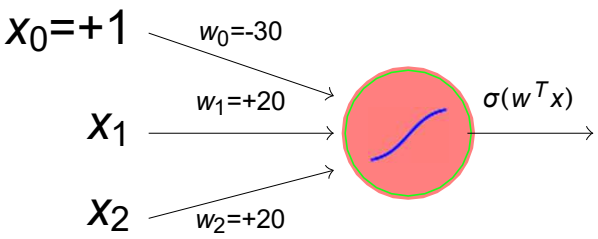
Consider a perceptron with output 0/1 as below



x_1	x_2	Output
0	0	0
0	1	

An Example

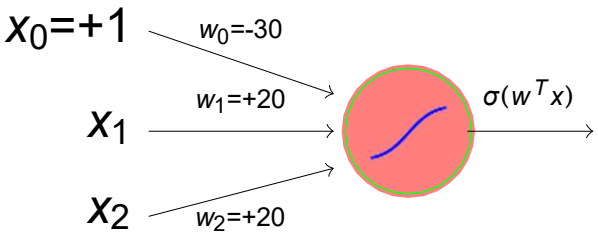
Consider a perceptron with output 0/1 as below



x_1	x_2	Output
0	0	0
0	1	0
1	0	

An Example

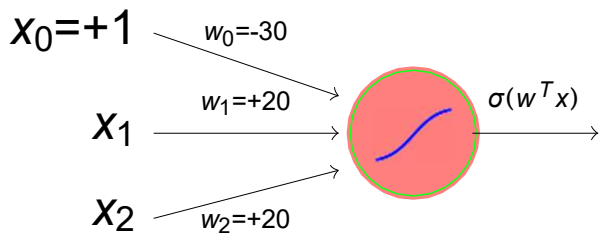
Consider a perceptron with output 0/1 as below



x_1	x_2	Output
0	0	0
0	1	0
1	0	0
1	1	

An Example

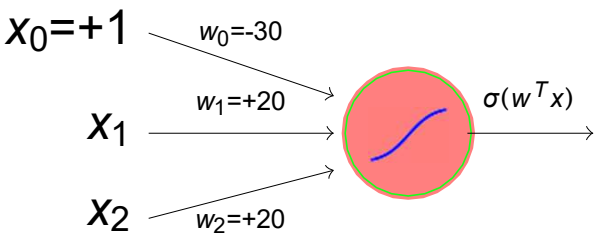
Consider a perceptron with output 0/1 as below



x_1	x_2	Output
0	0	0
0	1	0
1	0	0
1	1	1

An Example

Consider a perceptron with output 0/1 as below



x_1	x_2	Output
0	0	0
0	1	0
1	0	0
1	1	1

This perceptron computes logical AND

- w_0=-10 gives logical OR
- w_0=10, w_1=-20 with single input gives logical NOT
- XOR is not possible