Deep Learning: Theory and Practice

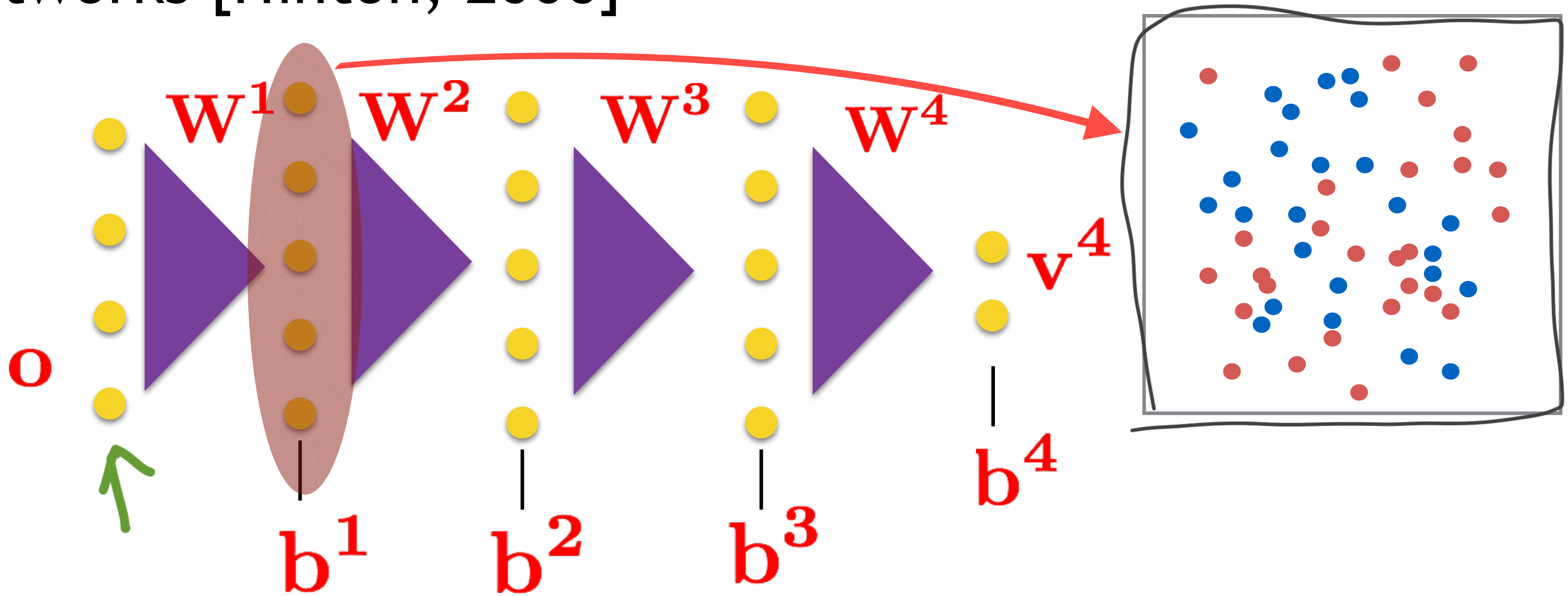
Deep Learning - Practical Considerations

02-04-2020

deeplearning.cce2020@gmail.com

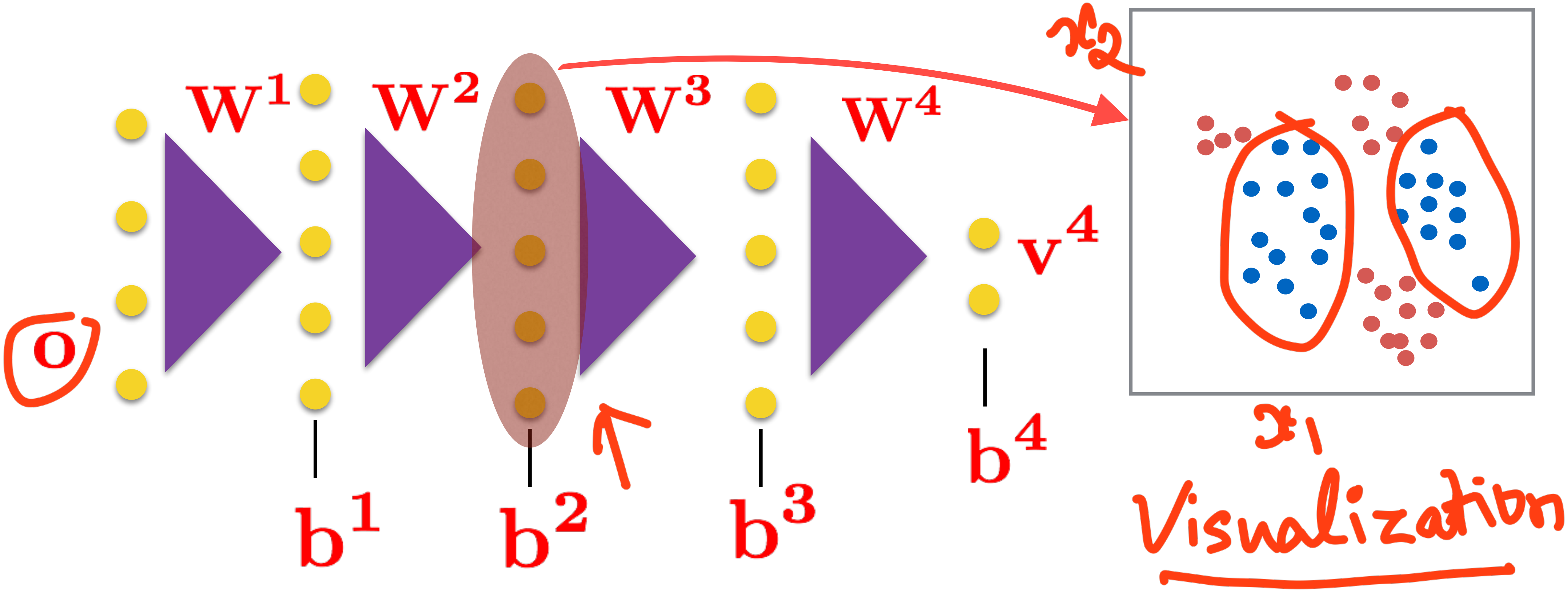
Deep Networks Intuition

Neural networks with multiple hidden layers - Deep networks [Hinton, 2006]



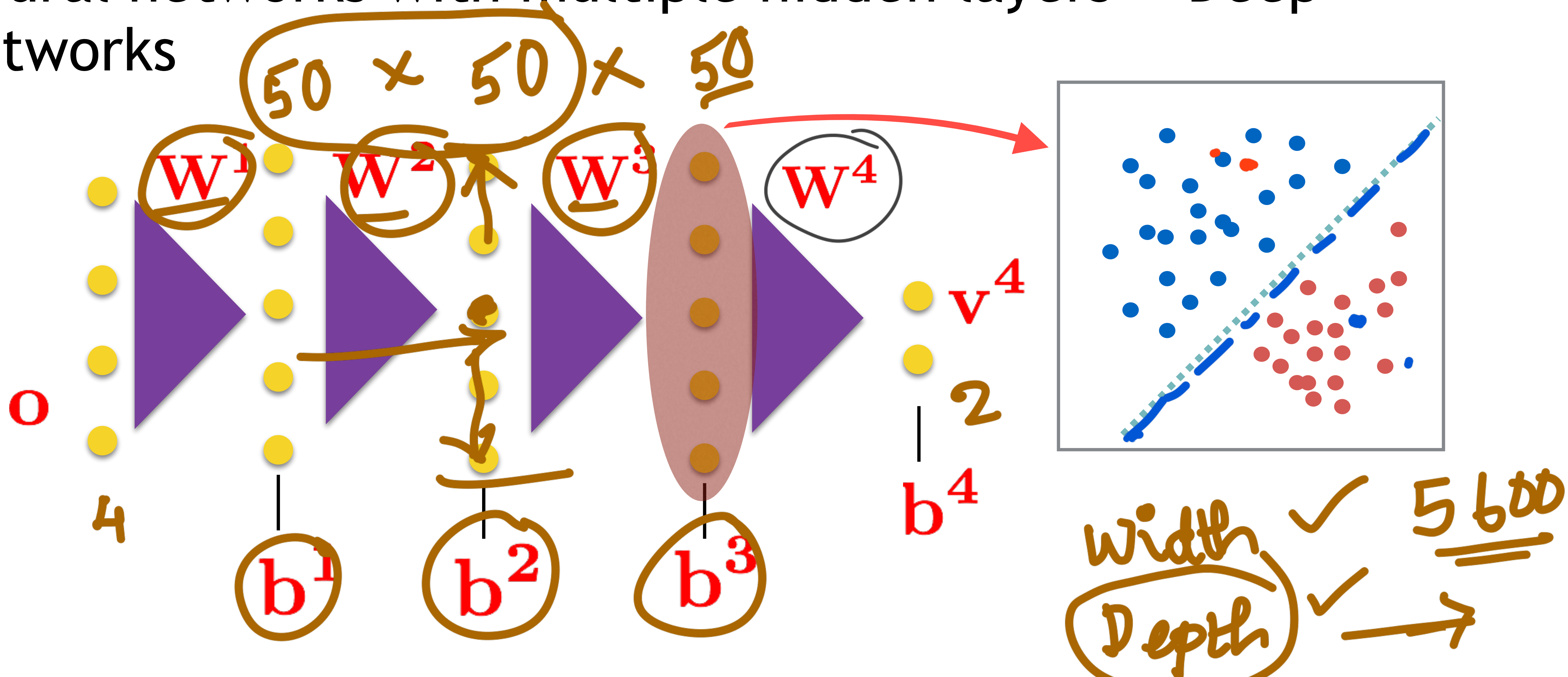
Deep Networks Intuition

Neural networks with multiple hidden layers - Deep networks



Deep Networks Intuition

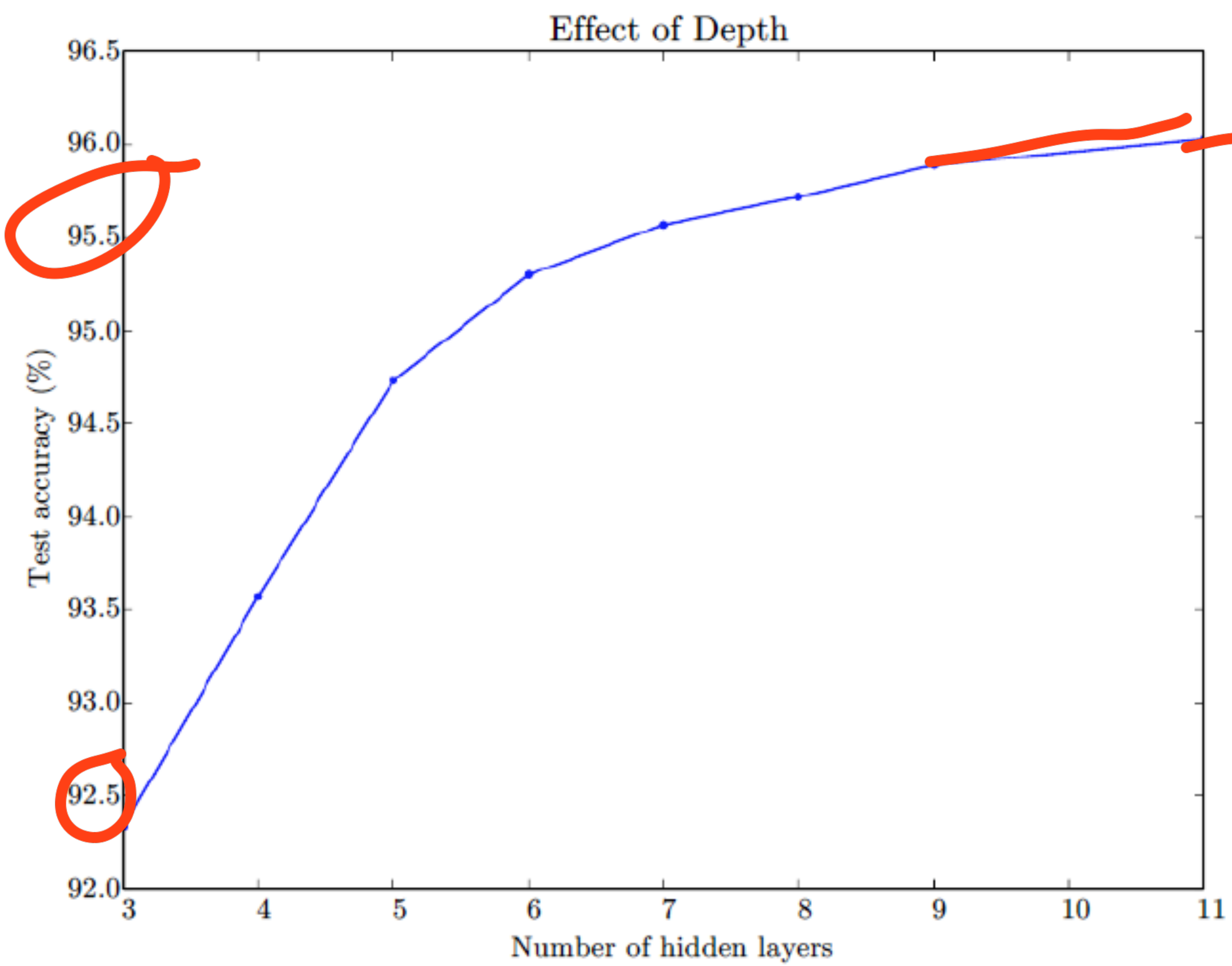
Neural networks with multiple hidden layers - Deep networks



Deep networks perform hierarchical data abstractions which enable the non-linear separation of complex data samples.

Established 1911

Need for Depth

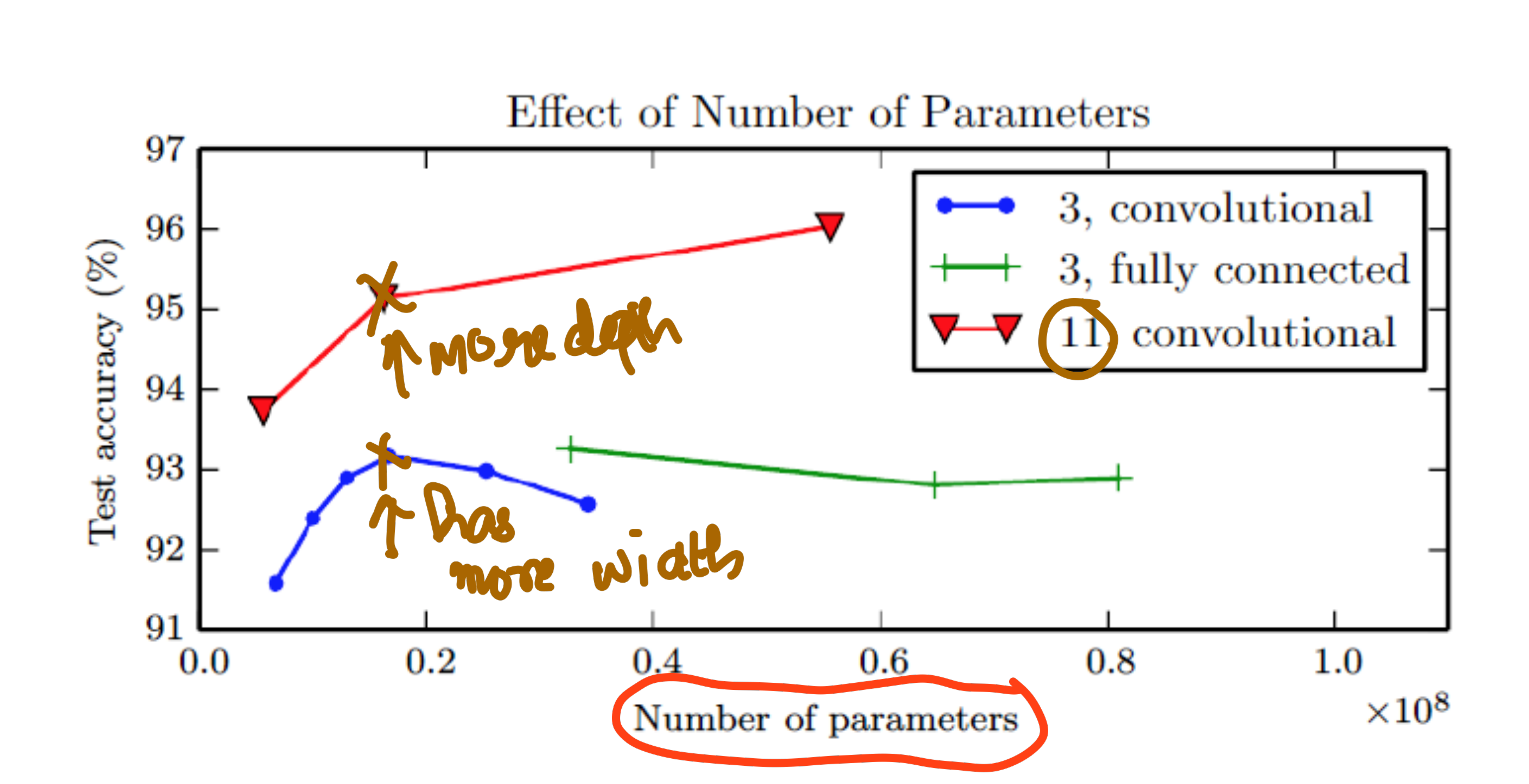


MNIST

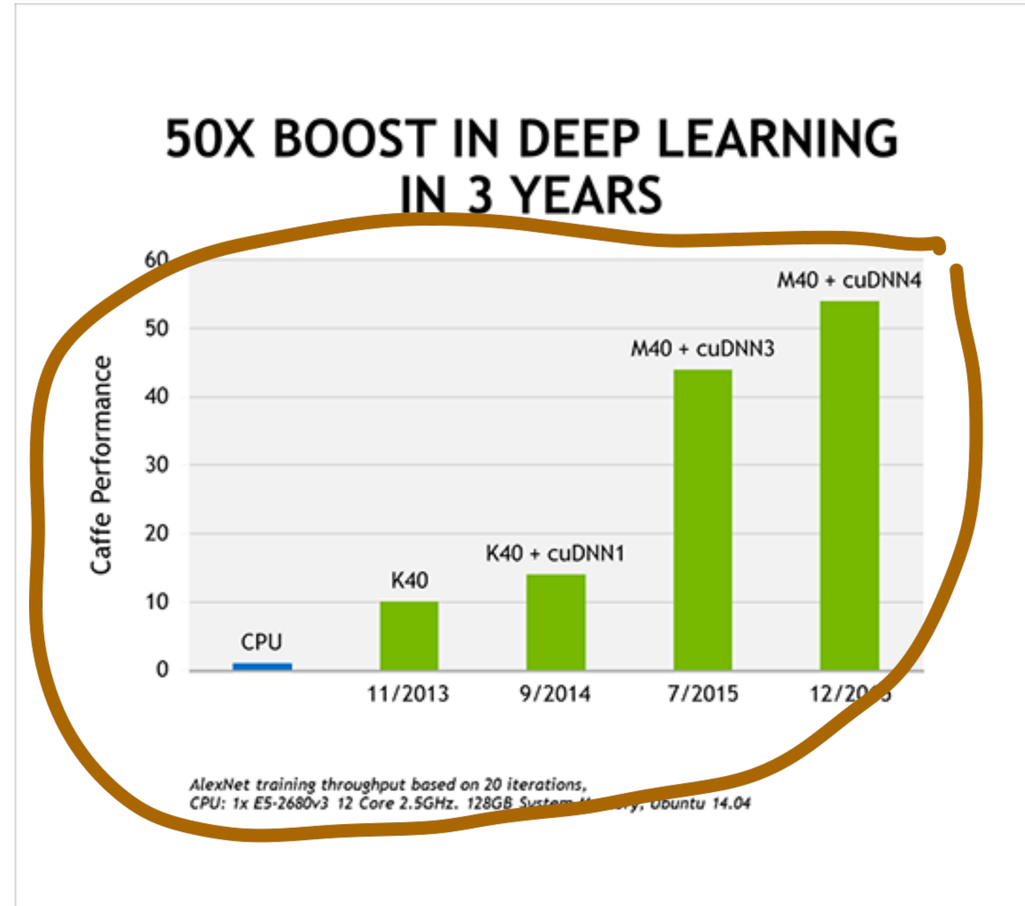
$$\boldsymbol{h}^{(1)} = g^{(1)}\left(\boldsymbol{W}^{(1)\top}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right)$$

$$\boldsymbol{h}^{(2)} = g^{(2)}\left(\boldsymbol{W}^{(2)\top}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right)$$

Need for Depth



Deep Networks



- Are these networks trainable ?

- Advances in computation and processing
- Graphical processing units (GPUs) performing multiple parallel multiply accumulate operations.
- Large amounts of supervised data sets

Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Deep Networks

- Will the networks generalize with deep networks ← Deviation from training
    - DNNs are quite data hungry and performance improves by increasing the data.
    - Generalization problem is tackled by providing training data from all possible conditions.
        - Many artificial data augmentation methods have been successfully deployed
    - Providing the state-of-art performance in several real world applications.

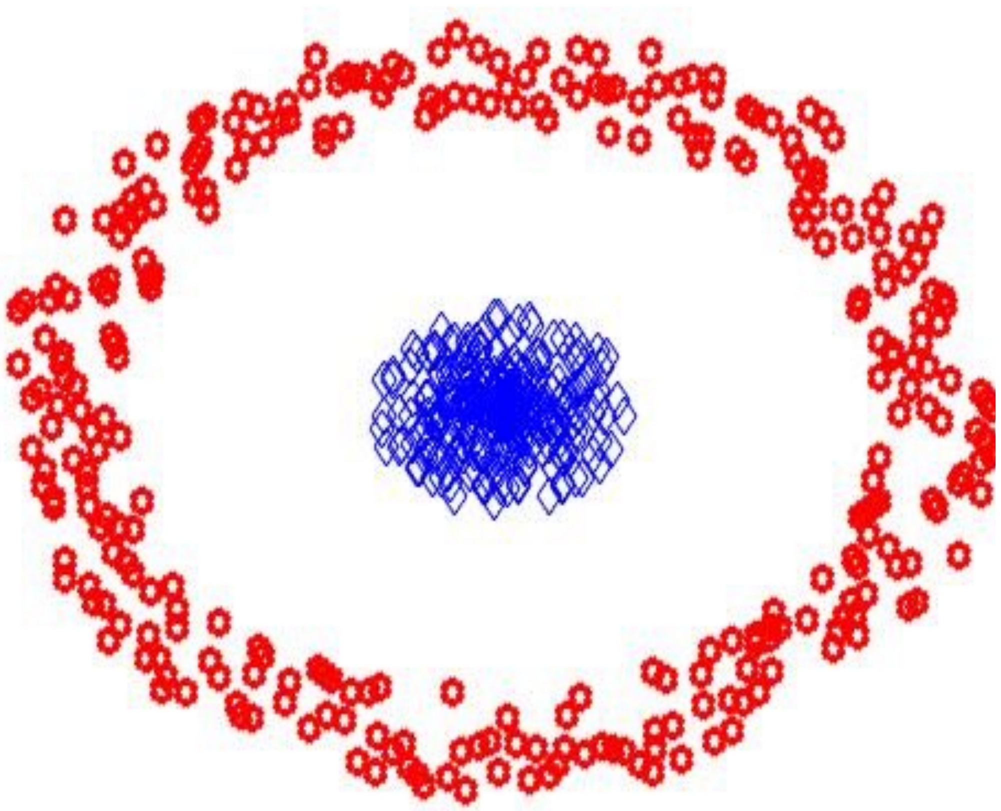
Established 1911

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Representation Learning in Deep Networks

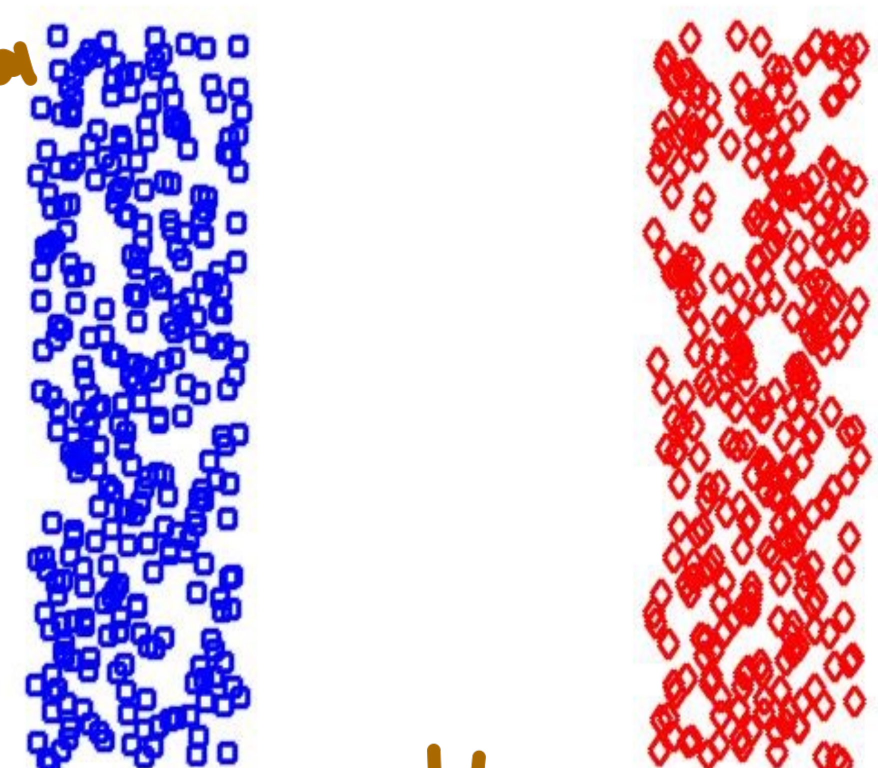
- The input data representation is one of most important components of any machine learning system. $\|y-t\|^2$

Cartesian Coordinates



Non linear

Polar Coordinates



$\|\log y - \log t\|^2.$

Established 1911

INDIAN INSTITUTE OF SCIENCE

भारतीय विज्ञान संस्थान

Representation Learning in Deep Networks

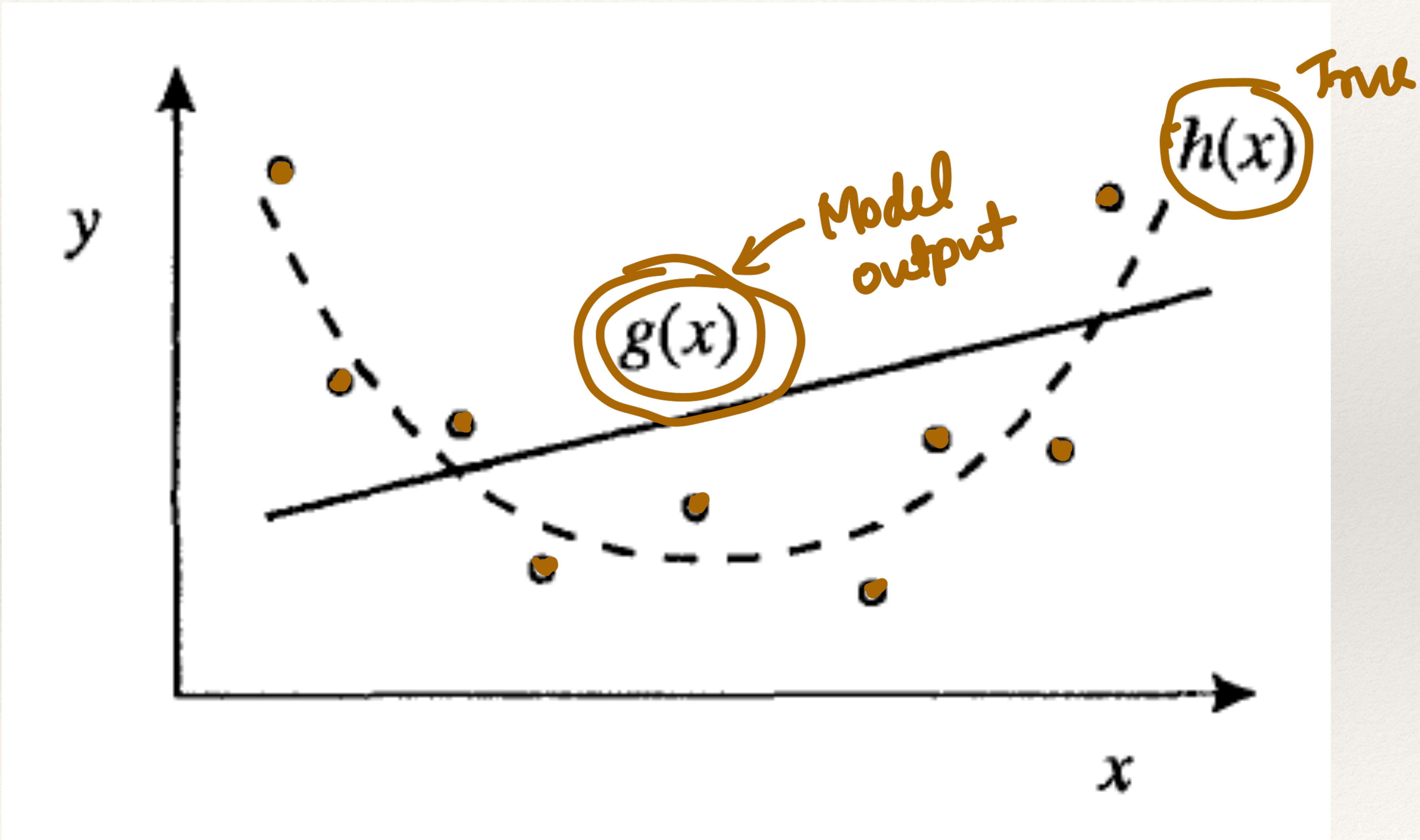
- The input data representation is one of most important components of any machine learning system.
  - Extract factors that enable classification while suppressing factors which are susceptible to noise.

- Finding the right representation for real world applications - substantially challenging.
  - Deep learning solution - build complex representations from simpler representations.
  - The dependencies between these hierarchical representations are refined by the target.

Established 1911

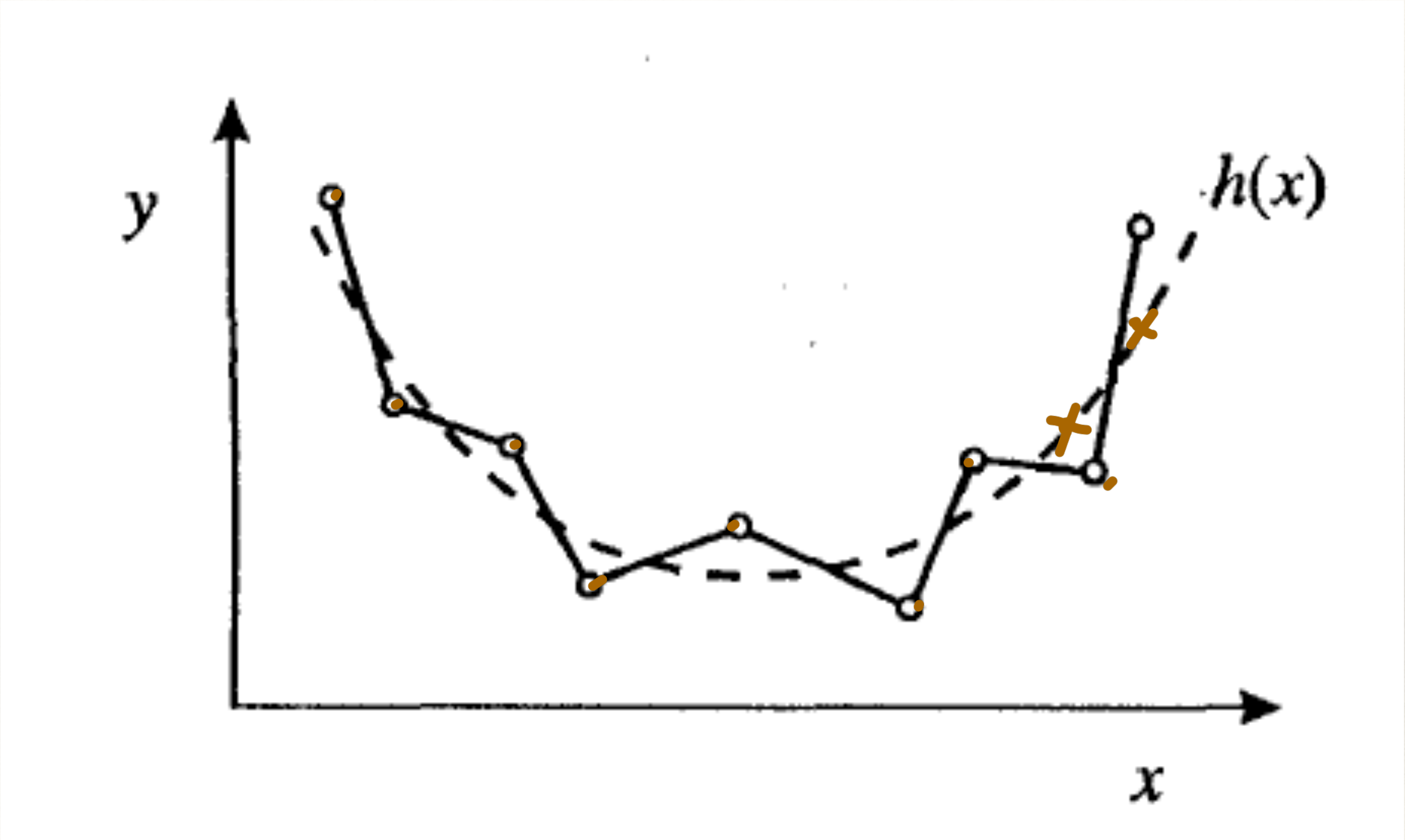
INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

Underfit



Overfit

→ Lack of Generalization.



Avoiding OverFitting In Practice

Regularization

$E_D(W)$ ← data dependent error function.

$$\|W^1\|_2^2 = \left(\sum_i \sum_j (W^1_{ij})\right)^2$$
$$+$$
$$\|W^2\|_2^2 = \sum\sum (W^2_{pq})$$

Overfitting ← following training data too closely.

Learn patterns

$E_W(w)$ ← Data independent loss function.
Model dependent

Back propagation

$$\boxed{E(W)} = E_D(W) + \lambda E_W(W)$$

λ ← Regularization Parameter.

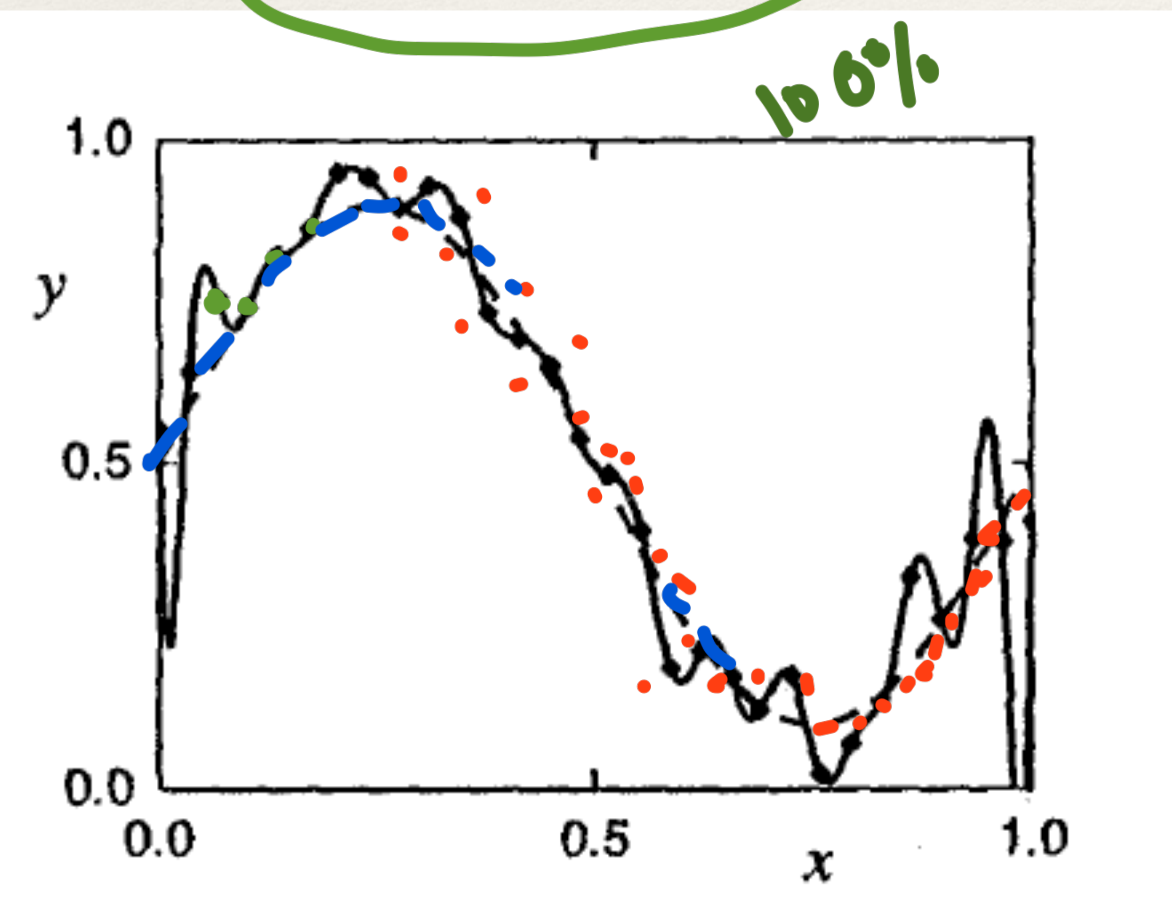
λ - small overfitting
λ - large (underfit)

Weight Decay Regularization $E_W(W) = \|W\|_2^2$

I. Weight Decay Regularization

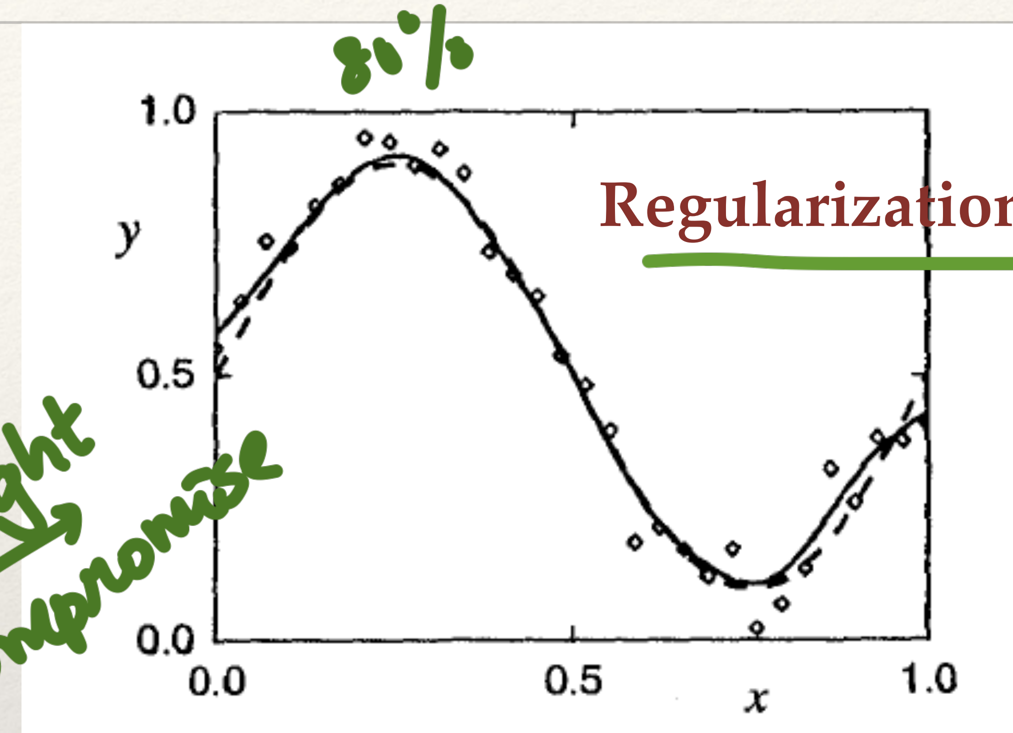
Regularization = 0

100%



80%

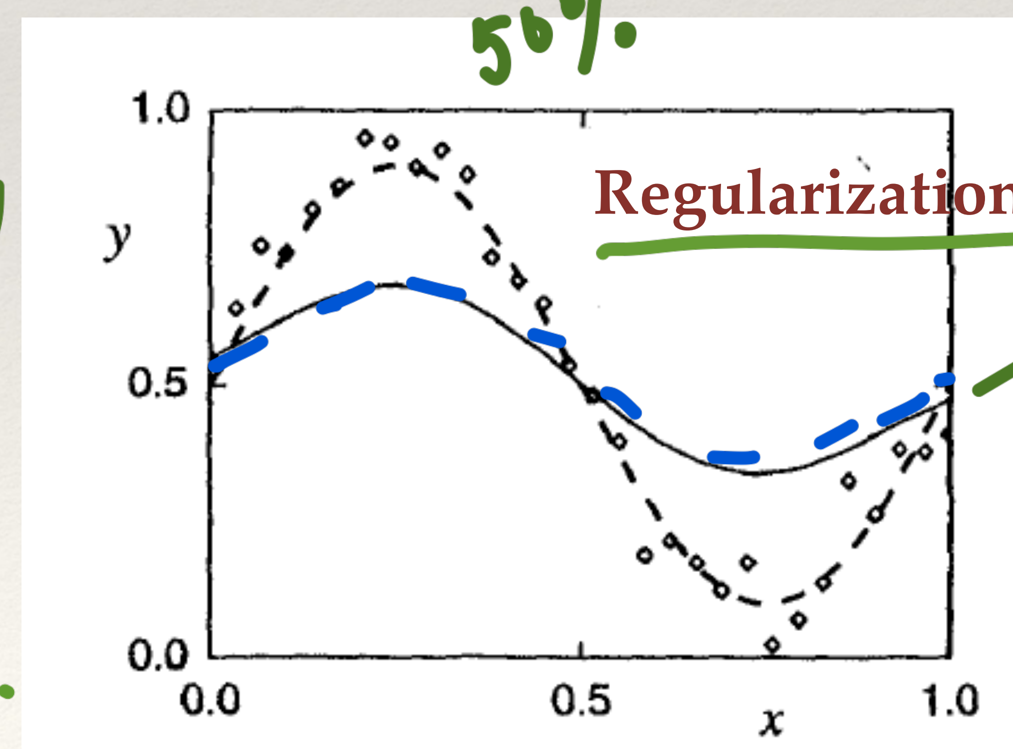
Regularization = 40



Right compromise

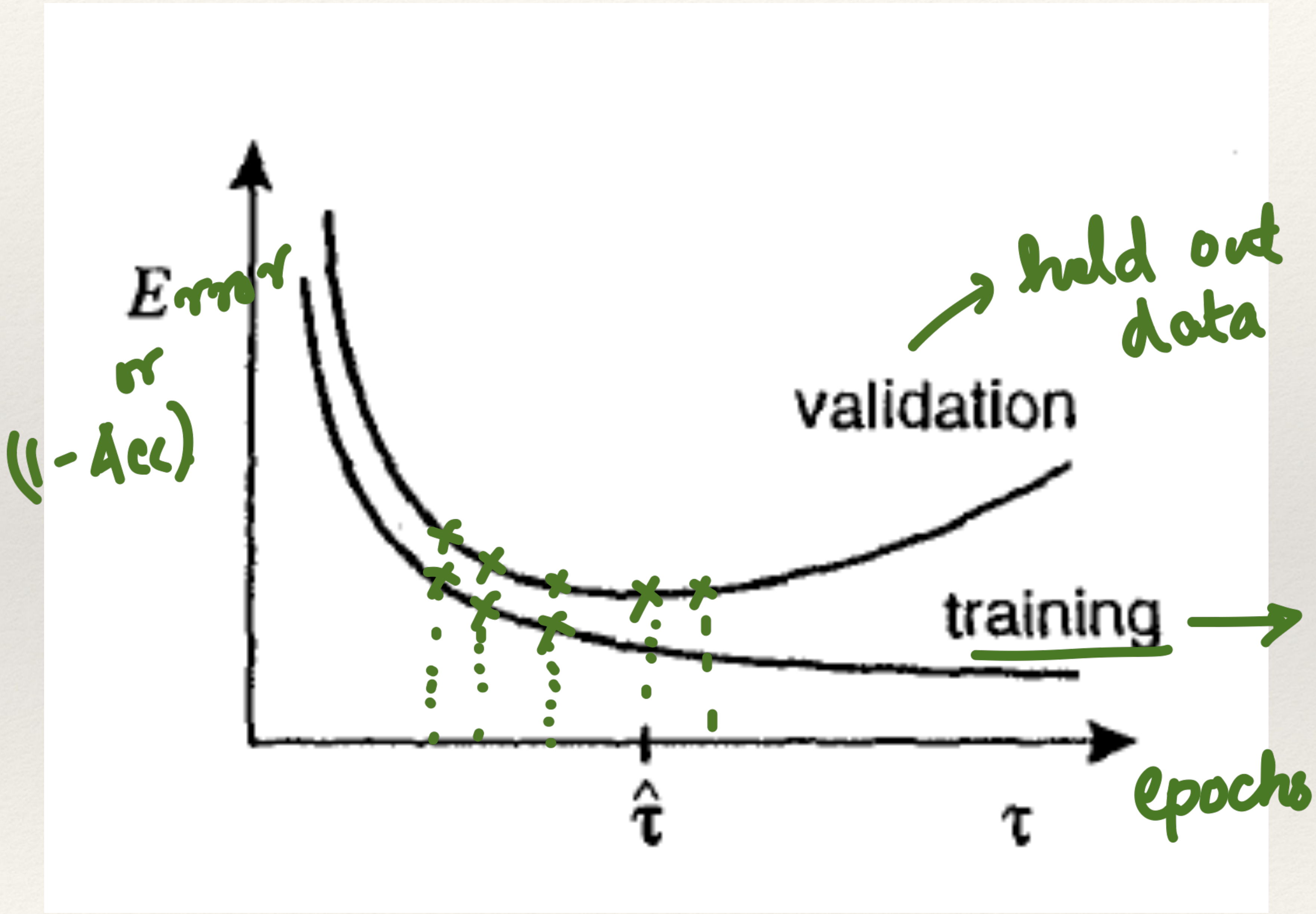
50%

Regularization = 4000



λ – small number gives you best compromise.

II. # Early Stopping



Most Popular in Practice

Stop training when validation performance does not improve.

Ⅲ. Batch Normalization

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

2015

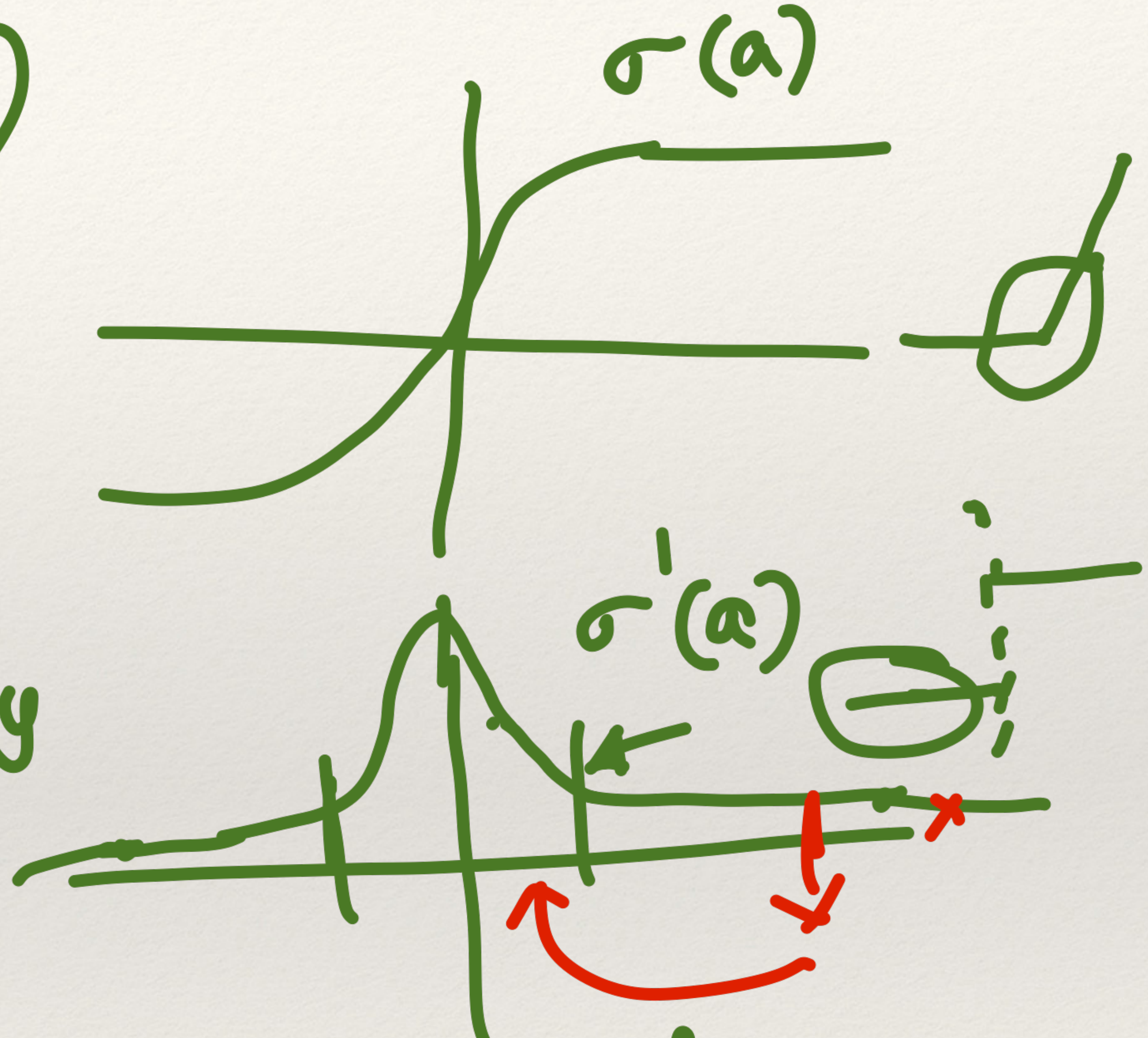
Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

Sigmoid as non-linearity (Two layers)

$$h^2 = \left(W^2 \sigma\underbrace{\left(W^1 \underline{x} + b^1\right)}_{z^1} + b^2\right)$$

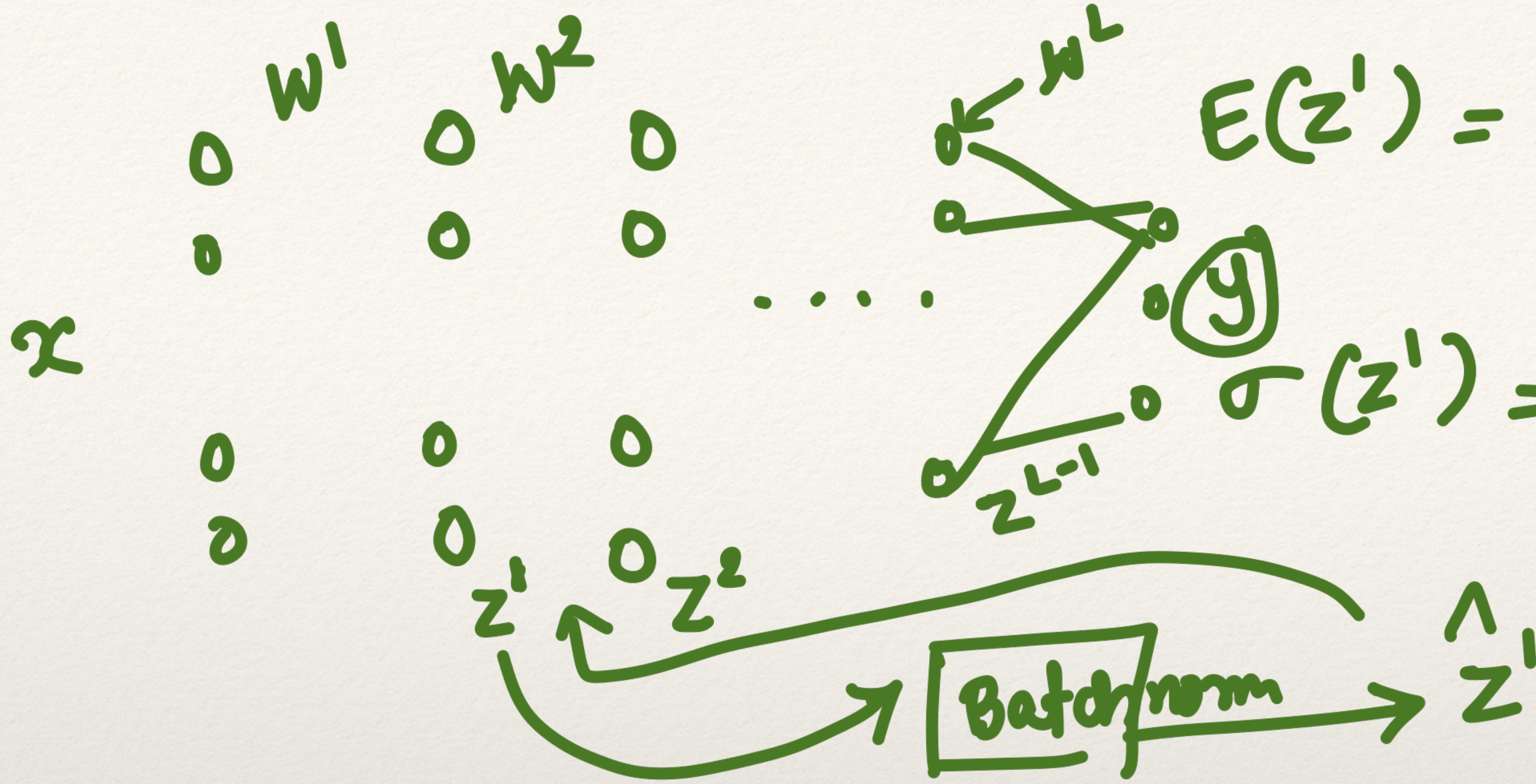
Model saturation if weight values or inputs are very high.



Batch normalization at every layer.

$$\underline{\underline{\hat{z}^1}} = \frac{z^1 - E(z^1)}{\sigma(z^1)}$$

← This makes each layer output normalized.
(close to 0, have small variance only).



$$E(z^1) = \frac{1}{N_b} \sum_{n \in N_b} z^1$$

$$\sigma(z^1) = \sqrt{\frac{1}{N_b} \sum_{n} \left(z^1 - E(z^1)\right)^2}$$

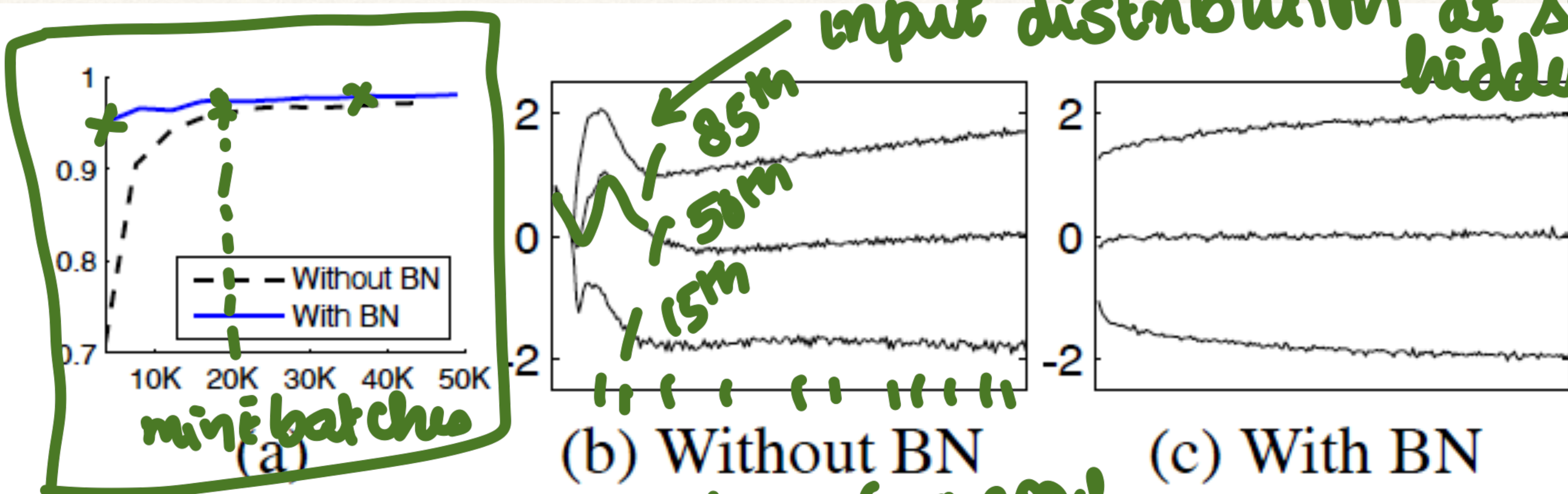
N_b - Mini Batch size should be $O(10^2)$

Can be done at every layer.

* higher learning rate
* Reduced number of epochs

Effect of Batch Normalization

input distribution at some hidden layer in terms of percentile



z'

$\{z'\}_{n=1}^{N}$

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

IV. # Dropout Strategy in Neural Network Training

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

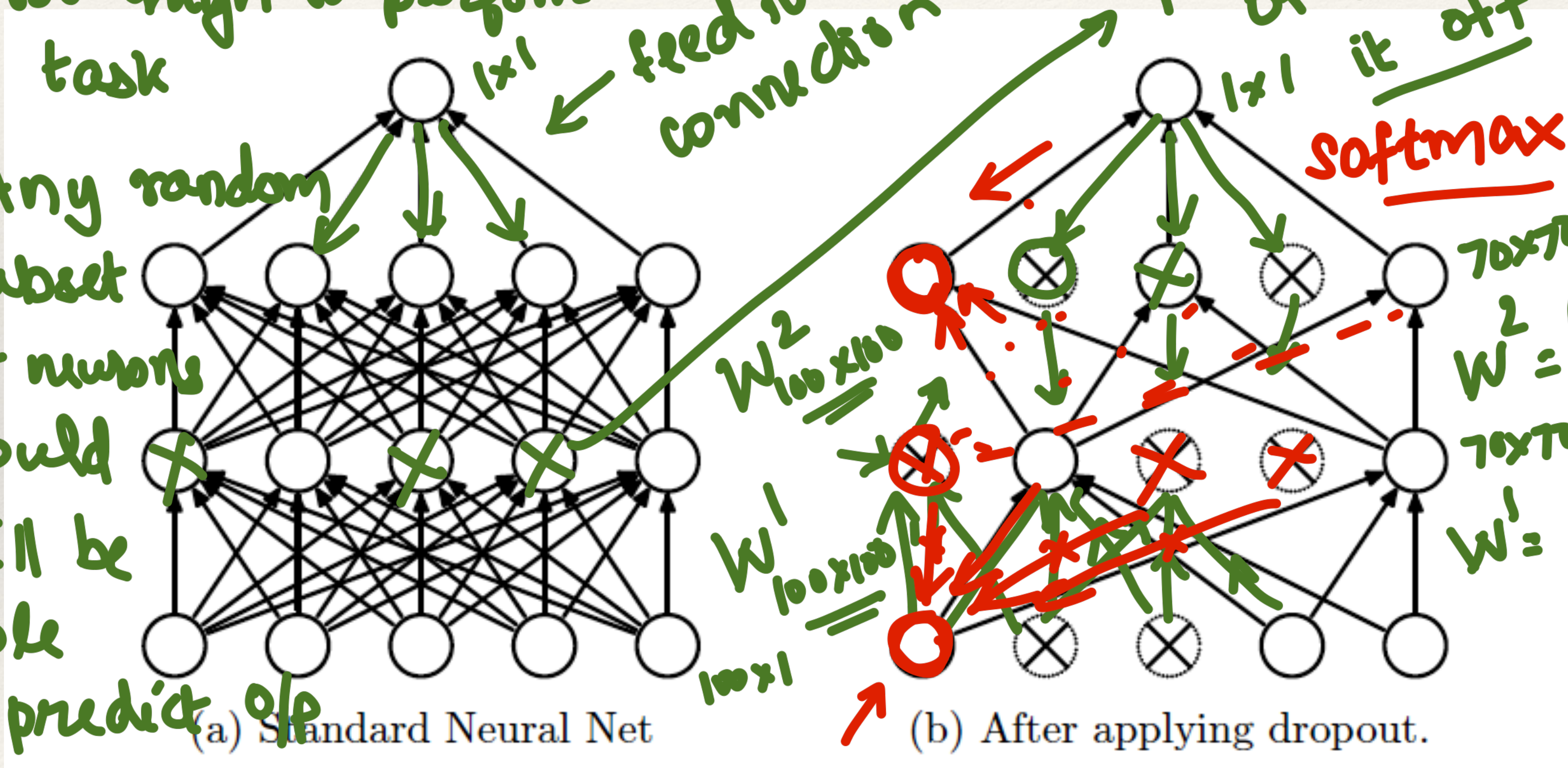
2013

Editor: Yoshua Bengio

Dropouts in Neural Networks

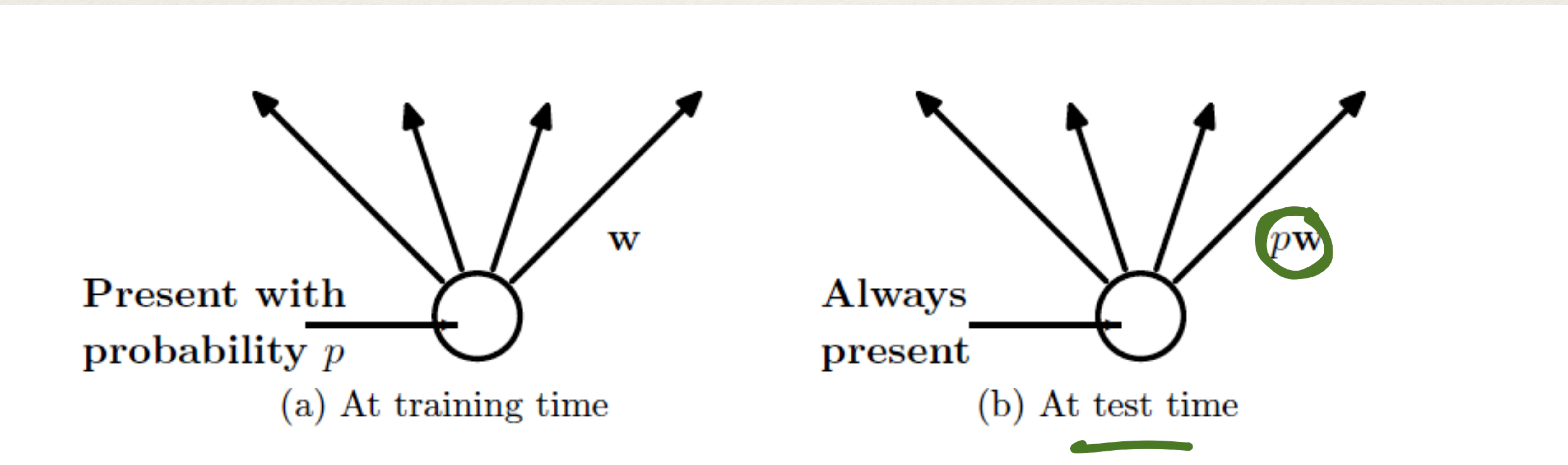
* Nodes (neurons) are too high to perform task

* Any random subset of neurons should still be able to predict o/p



Fixed Dropout probability (Sparsity)

Dropout in Training and Test



Consistent o/p
in testing

Dropout Application

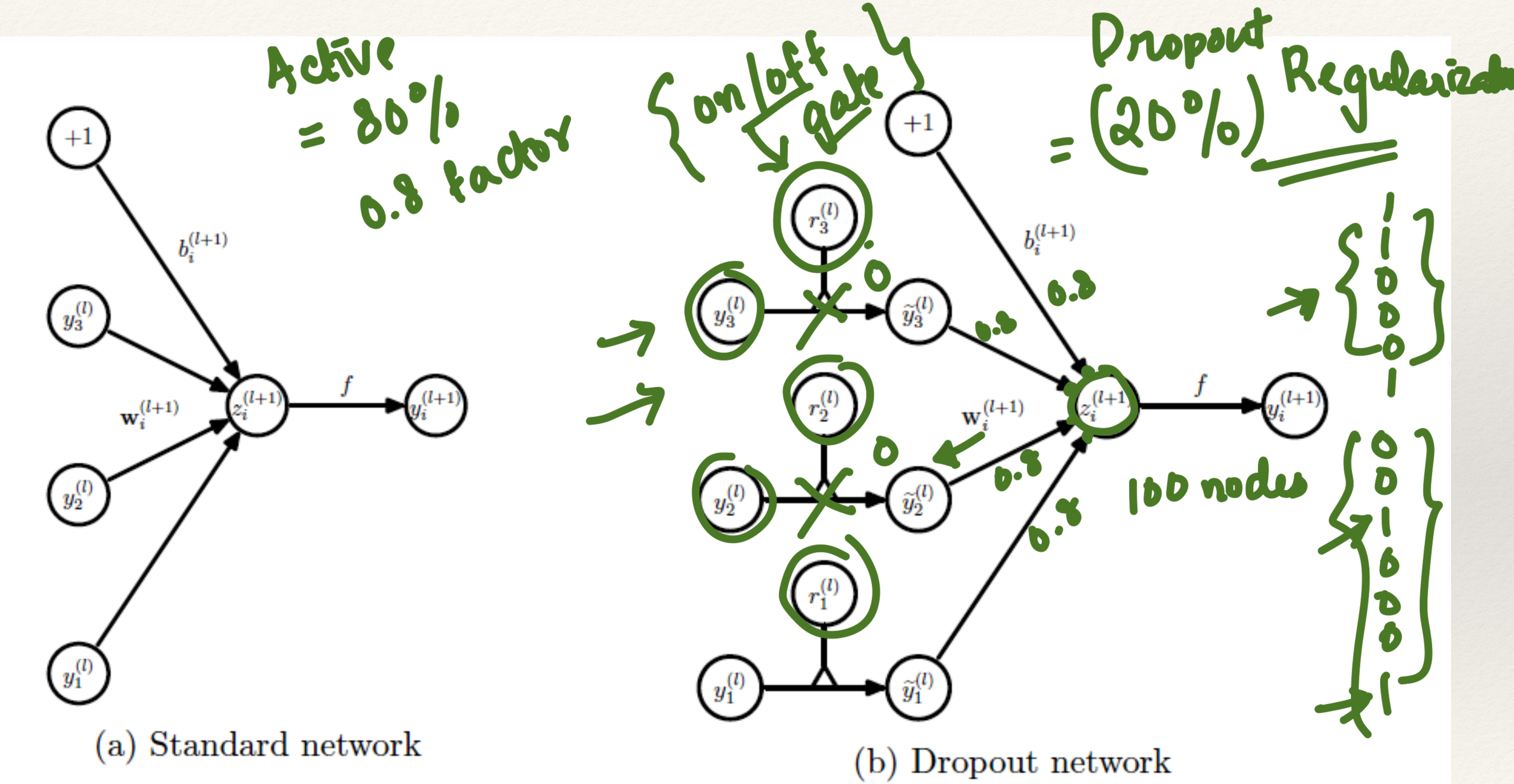


Figure 3: Comparison of the basic operations of a standard and dropout network.

Effect of Dropouts

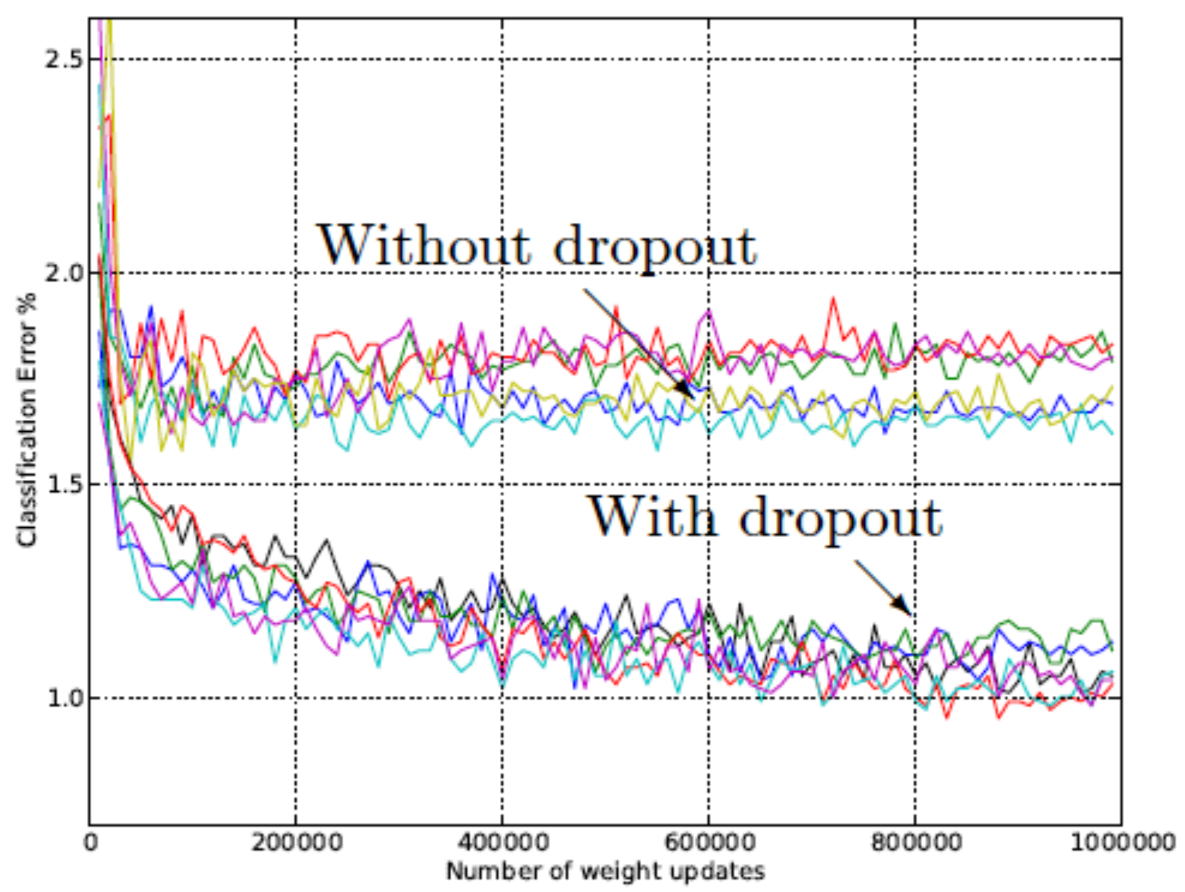
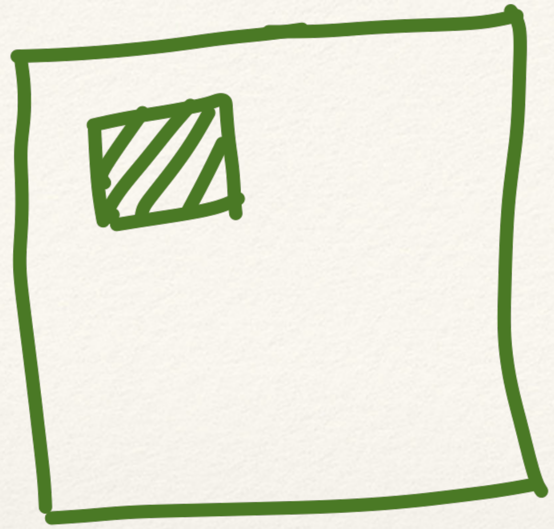


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Convolutional Neural Networks

CNNs



V/s $\longrightarrow$

DNN

weigh

$(Wx + b)$

1-D

2-D patterns

$\downarrow$ Can I also have weights which are 2-D and local. [process parts of Image].

Convolution Operations

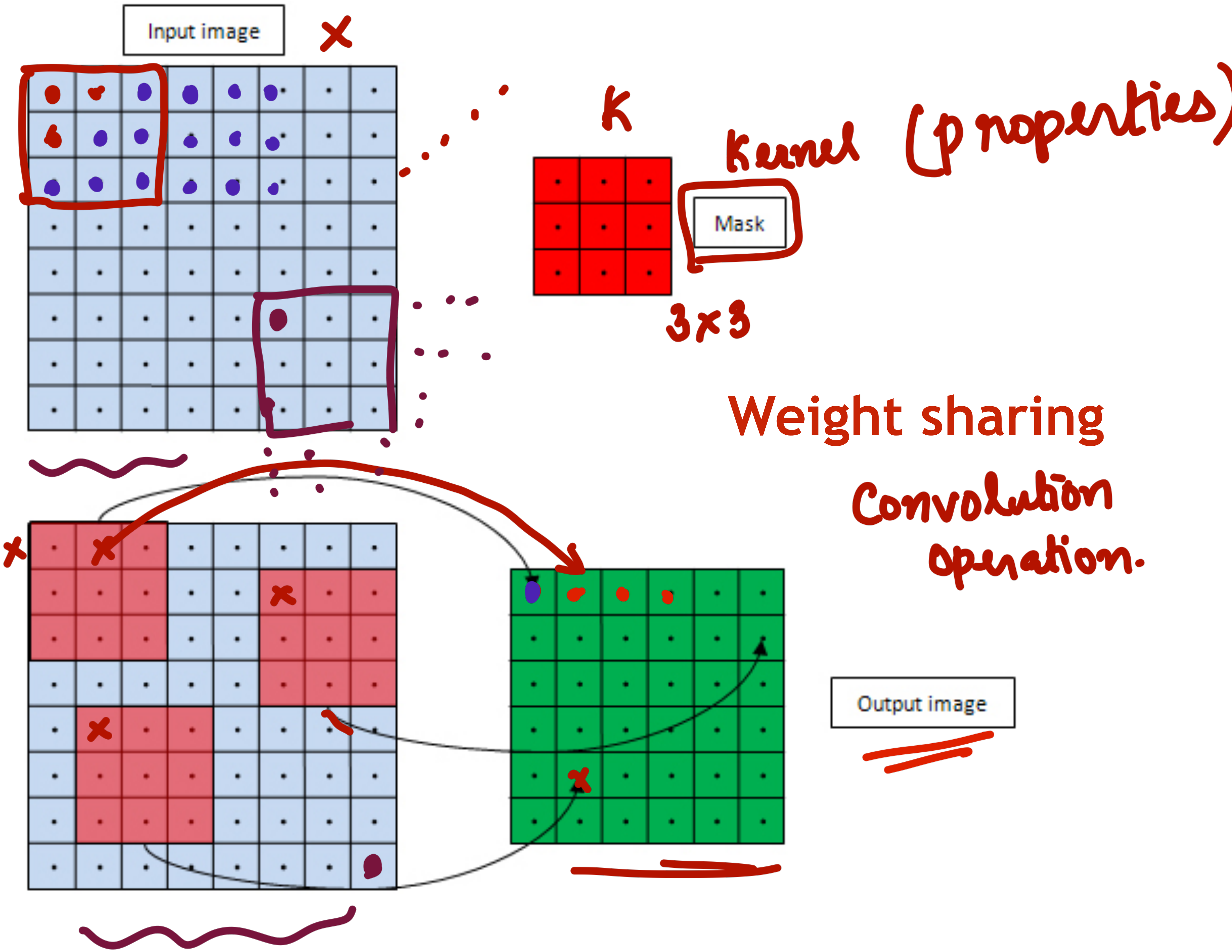
$p = 1 \ldots P$

$$Y^{p}(i,j) = \underbrace{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1}}_{K \in \mathbb{R}^{M\times N}} \underbrace{X(i+m, j+n)\, \boxed{K^{p}(m,n)}}$$

Learnable parameter

Destroys 2-D structure

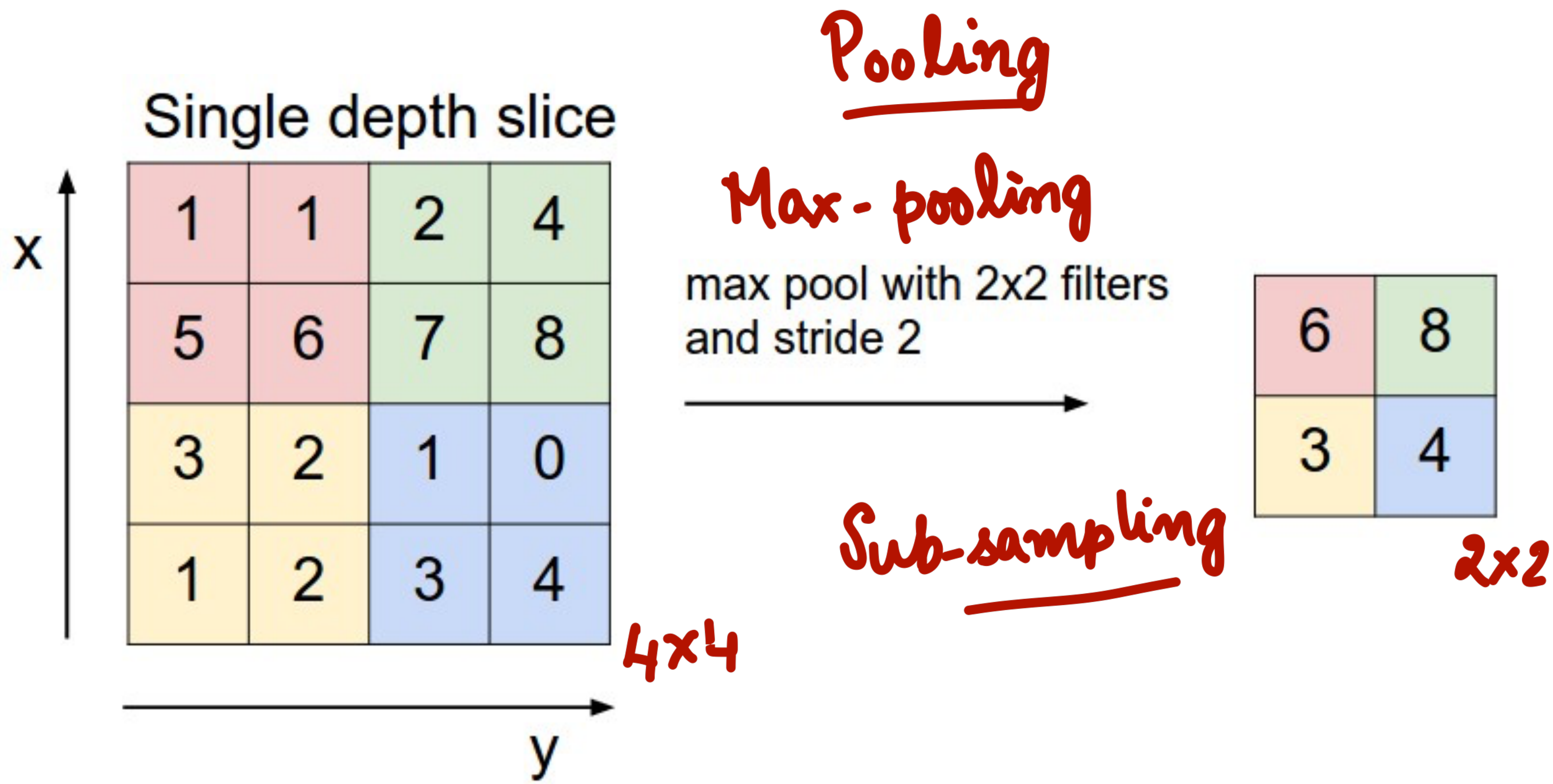
Other Architectures - Convolution Operation

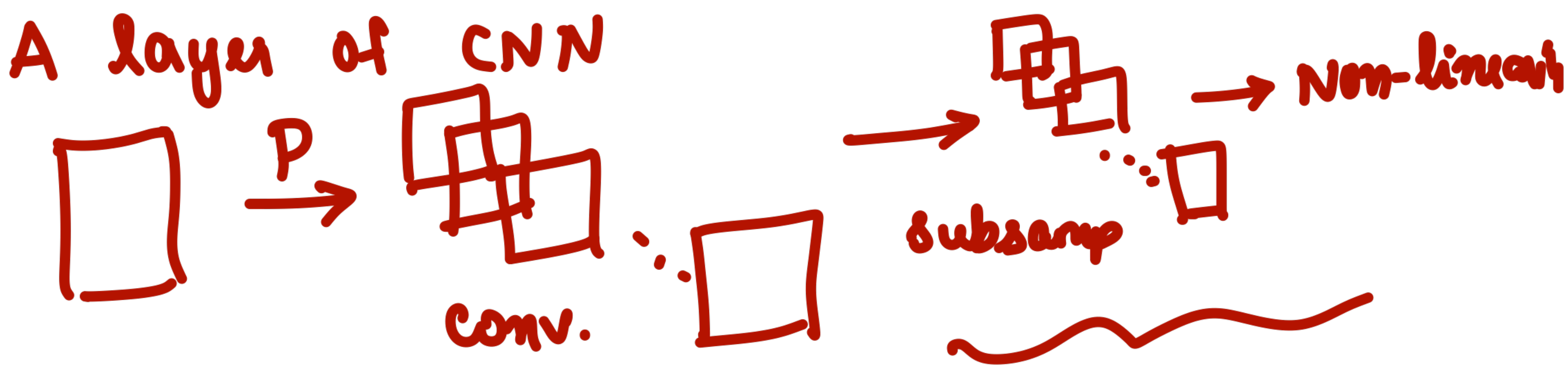


Weight sharing

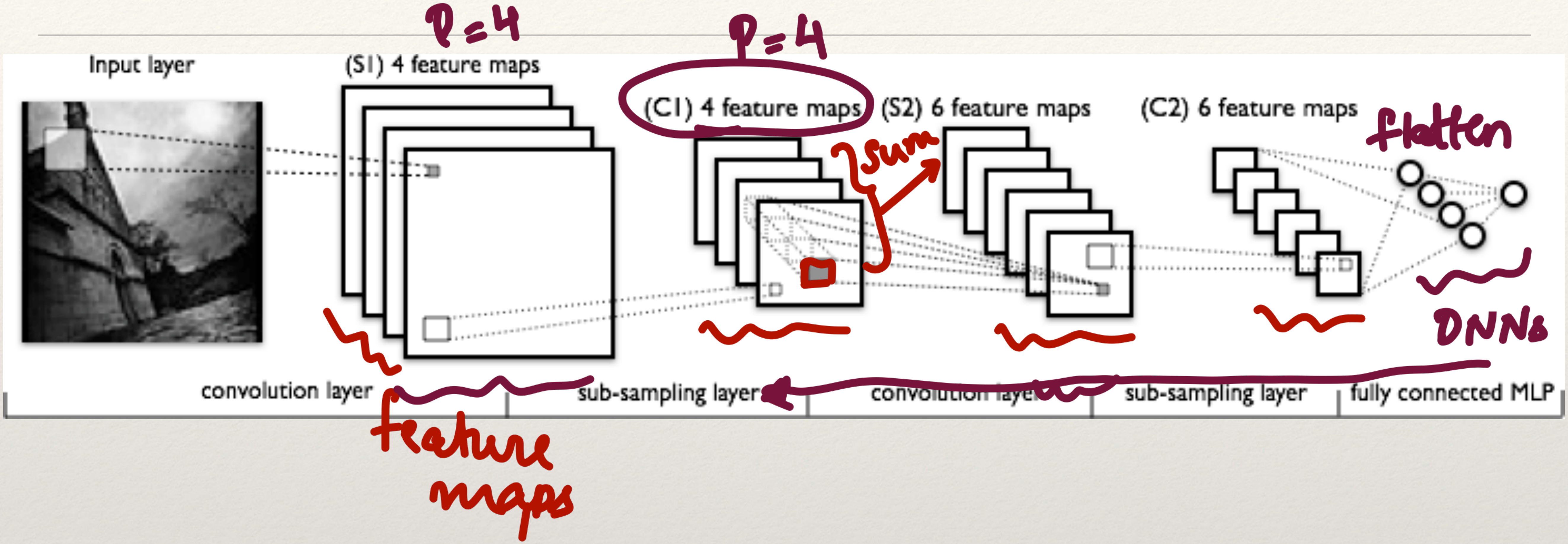
Convolution operation.

Max Pooling Operation



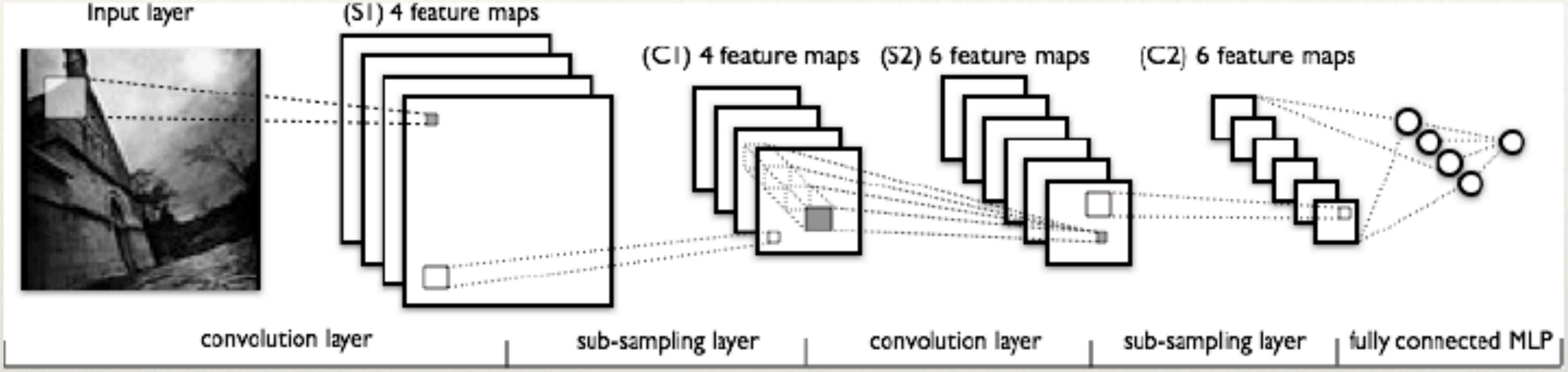


Convolutional Neural Networks



- Multiple levels of filtering and subsampling operations.
- Feature maps are generated at every layer.

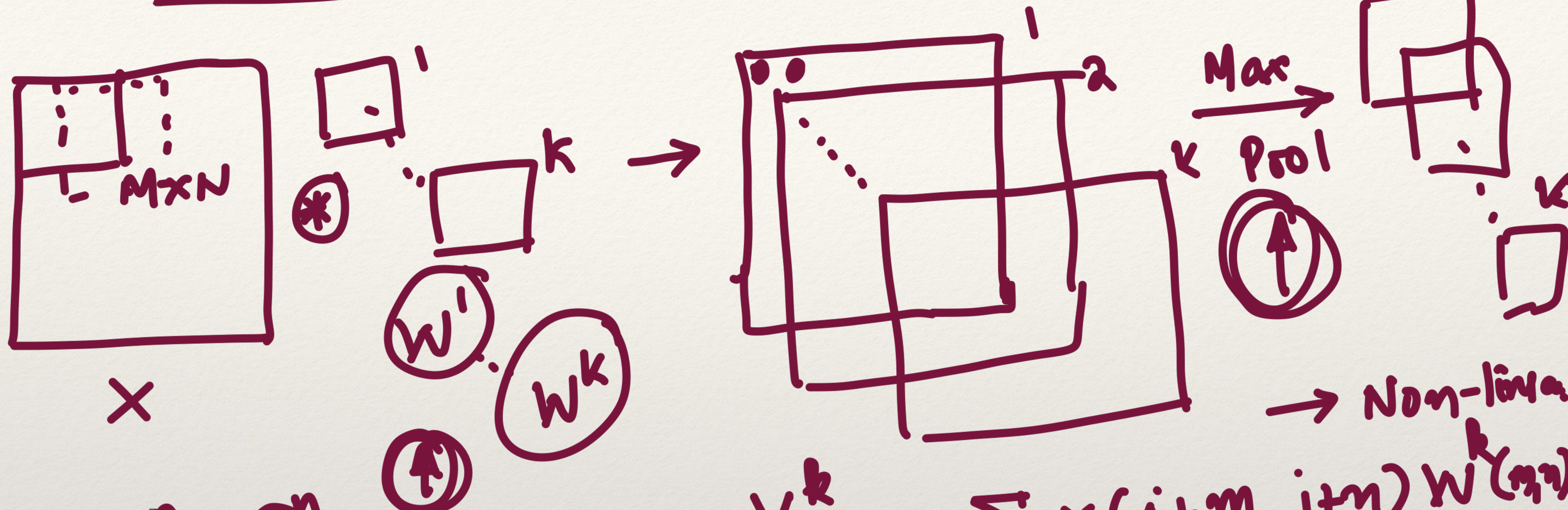
Convolutional Neural Networks



- Multiple levels of filtering and subsampling operations.
- Feature maps are generated at every layer.

Back Propagation in CNNs

2-D CNN (Convolution)



$$E = \sum_n E\left[\underline{t}^n, y^{L^n}\right]$$

$$Y^k = \sum_{m,n} x(i+m, j+n)\, W^k(m,n)$$

$$[W^k]$$

$$\frac{\partial E}{\partial W^k}$$

$$\frac{\partial E}{\partial y^k} \longrightarrow \frac{\partial E}{\partial W^k}$$

$$(M+1, N+1)$$

$X - \mathbb{R}^{D_1 \times D_2}$ If no padding $Y^k \in \mathbb{R}^{D_1 - M,\, D_2 - N}$

$$y^k[0,0] = x[0,0]\, w^k[0,0] + x[0,1]\, w^k[0,1] + \dots + x[0,N]\, w^k[0,N] + x[1,0]\, w^k[1,0] + \dots\, x[M,N]\, w^k[M,N]$$

Convolution

$$y^k[0,1] = x[0,1]\, w^k[0,0] + \dots + x[0,N+1]\, w^k[0,N] + x[1,1]\, w^k[1,0] + \dots + x[M+1,N]\, w^k[M,N]$$

$[w^k]$

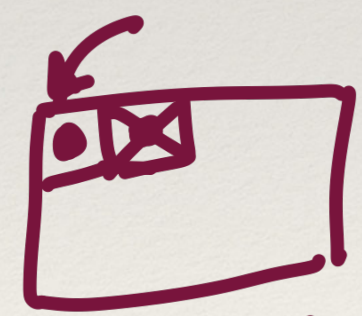
$M+1,\ N+1$

scalar

$$\left[\frac{\partial E}{\partial y^k}\right] \longrightarrow \left[\frac{\partial E}{\partial w^k}\right]$$

$(D_1 - M,\ D_2 - N)$

$M+1,\ N+1$



$M+1,\ N+1$

$$\frac{\partial \mathcal{E}}{\partial w^k[0,0]} = x[0,0] \cdot \frac{\partial \mathcal{E}}{\partial y^k[0,0]} + x[0,1] \cdot \frac{\partial \mathcal{E}}{\partial y^k(0,1)} + \dots + x[D_1-M, D_2-N] \cdot \frac{\partial \mathcal{E}}{\partial y^k[D_1-M, D_2-N]}$$

(*)

$$\frac{\partial \mathcal{E}}{\partial w^k[0,1]} = x[0,1] \cdot \frac{\partial \mathcal{E}}{\partial y^k[0,0]} + \dots + x[D_1-M, D_2-N+1] \cdot \frac{\partial \mathcal{E}}{\partial y^k[D_1-M, D_2-N]}$$

$\vdots$

M, N

D_1, D_2

$$\frac{\partial \mathcal{E}}{\partial w^k[m,n]} = \sum_{p,q=0,0}^{D_1-M,\,D_2-N} x[m+p, n+q] \frac{\partial \mathcal{E}}{\partial y^k[p,q]}$$

conv.

1 - layer CNN

Forward

$$X \circledast W^k \longrightarrow Y^k \xrightarrow[\text{pooling}]{\text{Max}} Z^k \xrightarrow{\text{ReLU}} R^k$$

bias

element wise

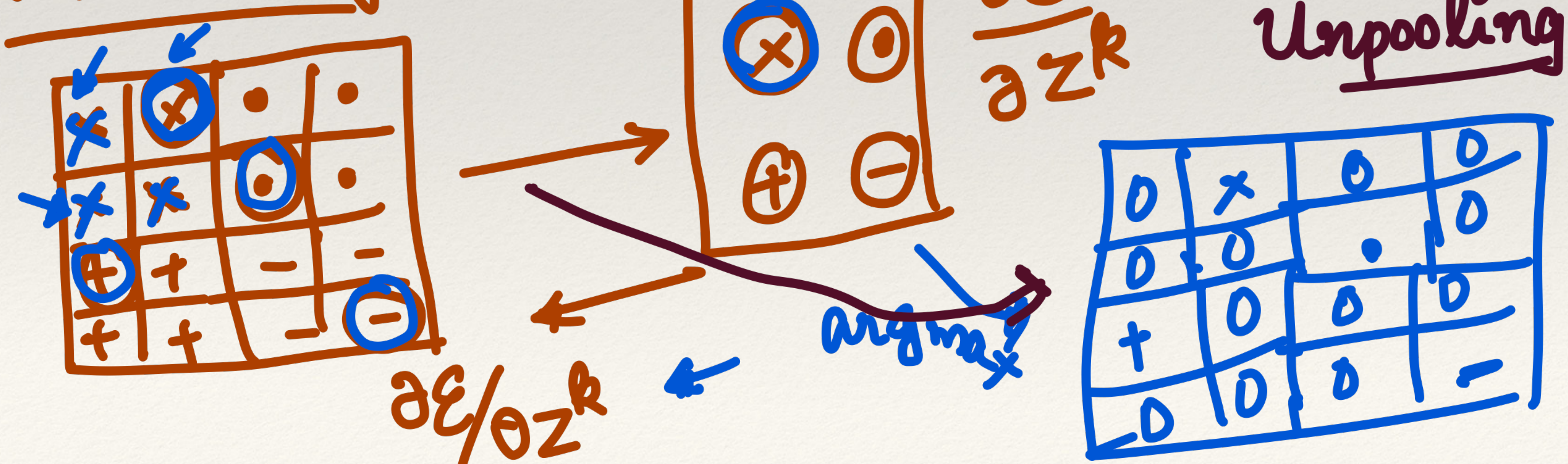
Backward

$$X \circledast \frac{\partial E}{\partial Y^k} \longrightarrow \frac{\partial E}{\partial W^k}$$

$y = Ax$

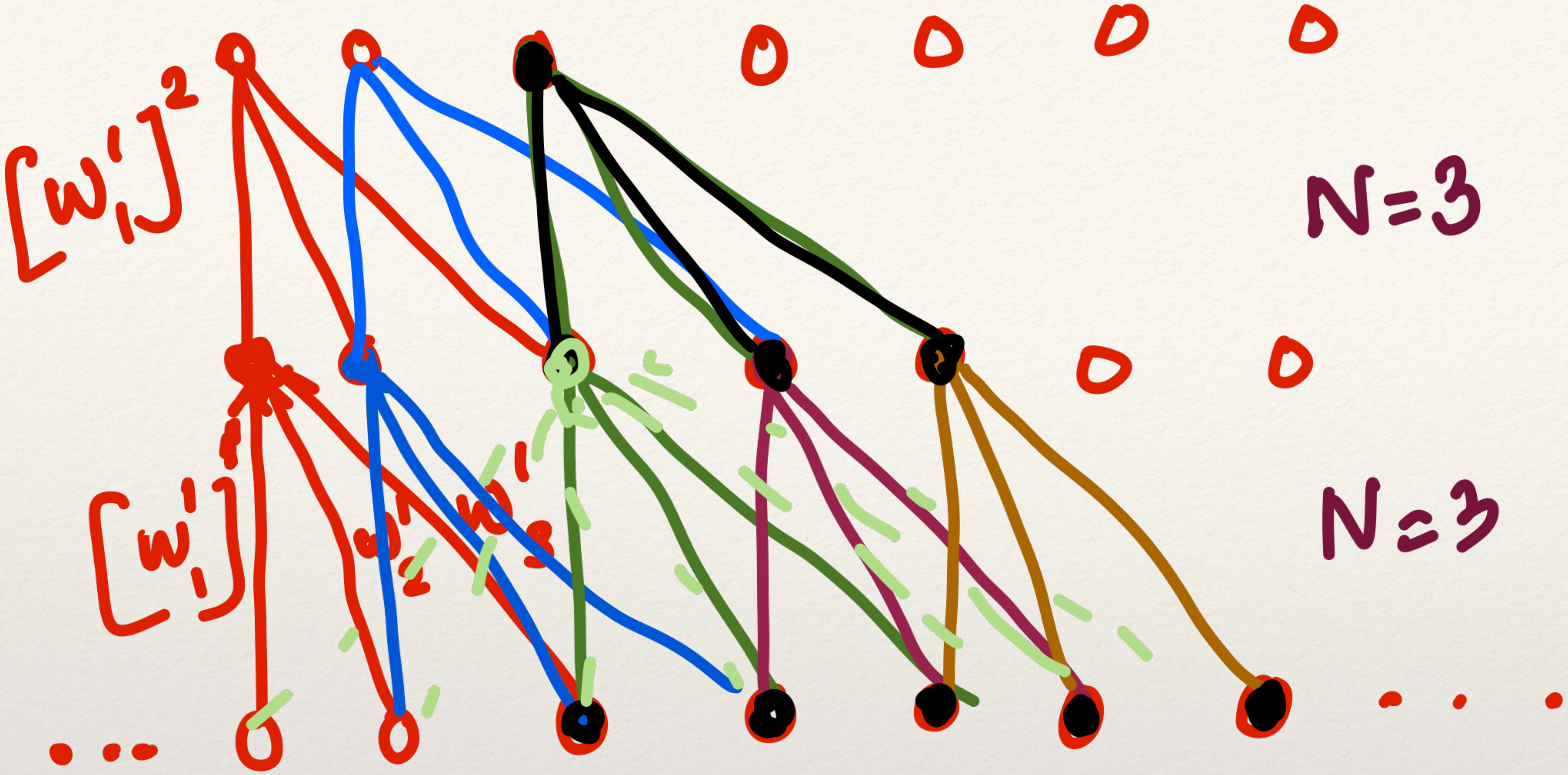
$$\frac{\partial}{\partial x}(Ax) = A^T$$

Max Pooling



1-D Convolutional Network

2nd layer



$x[n]$

1 kernel output

K-such outputs

K.

w_i^k → weight for i^{th} lag and k^{th} kernel

Layer in more depth are looking at more and more global properties of the input

$$\underline{w}^1 = \begin{bmatrix} 1/2 \\ -1 \\ 2 \end{bmatrix}$$

$$\underline{w}^2 = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \quad \ldots$$

Number of parameters

Typical values

N = 3 or 5 or 7

K = 64 or 128 or 256

CNN layers have much smaller # of parameters

→ Weight sharing

DNN

H

256

D

$W^{H \times D}$

D = 100

25600

CNN

K×N

$W^{K \times N}$

~2000

Expression for a 1-D CNN

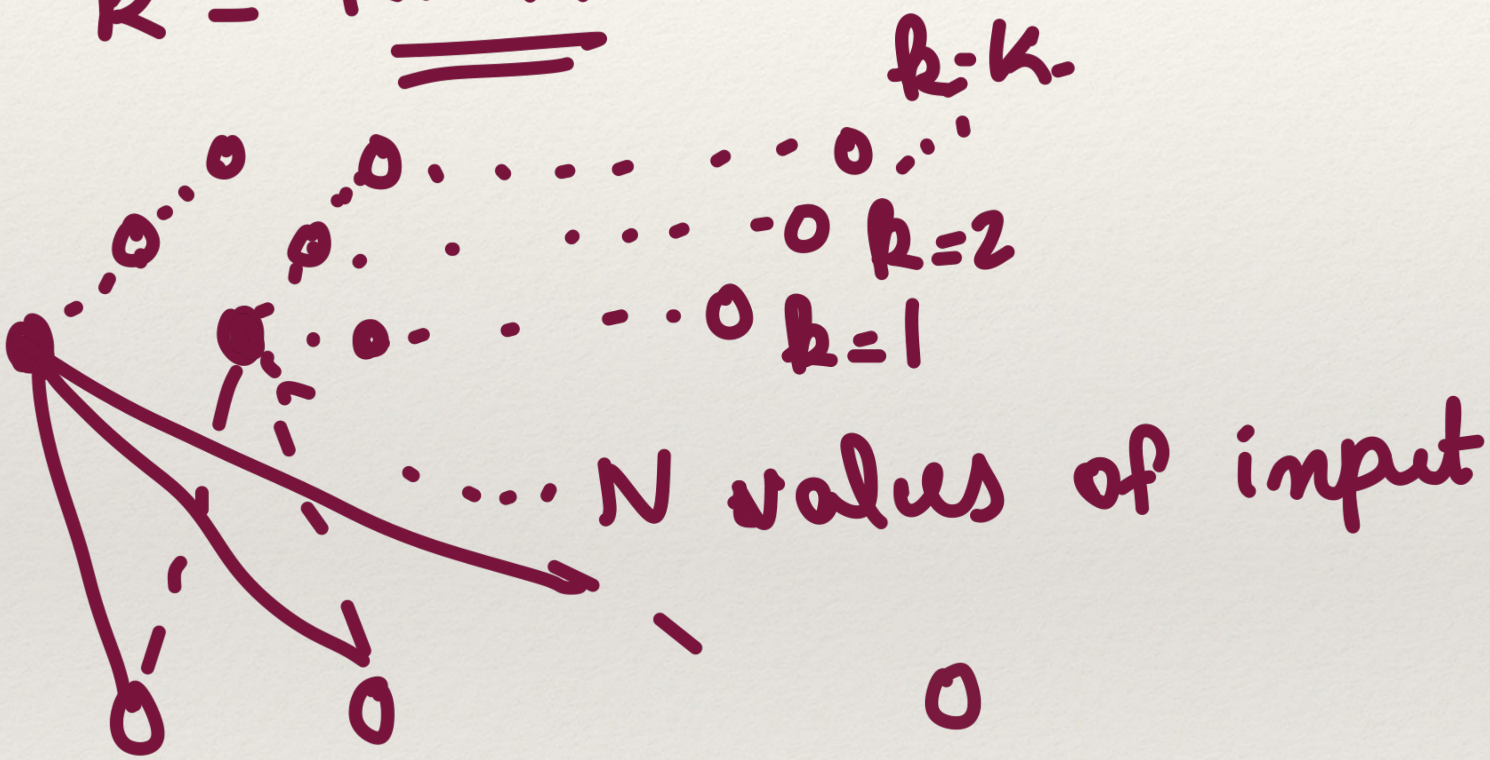
$$\boxed{y^k[n]} = \sum_{i=1}^{N} \underline{\underline{x[n+i]}}\, w_i^k$$

4

N - kernel width

$k = 1 \ldots \underline{\underline{K}}$

K - # kernels



parameters is $\underline{\underline{K \times N}}$

$$\underline{w^1} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \overset{\underline{w^2}}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}$$