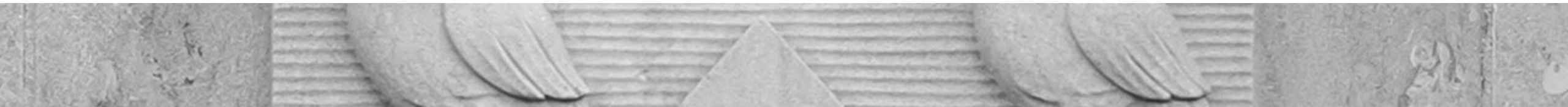


Deep Learning for Vision & Language

Computer Vision: Introduction and CNNs



RICE UNIVERSITY



Supervised Learning - Classification

Training Data



cat



dog



cat

.

.

.



bear

Test Data



.

.

.



Supervised Learning - Classification

Training Data

$$x_1 = [\text{] \quad y_1 = [\text{cat}]$$

$$x_2 = [\text{] \quad y_2 = [\text{dog}]$$

$$x_3 = [\text{] \quad y_3 = [\text{cat}]$$

•
•
•

$$x_n = [\text{] \quad y_n = [\text{bear}]$$

Supervised Learning - Classification

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}]$	$y_1 = 1$	$\hat{y}_1 = 1$
$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}]$	$y_2 = 2$	$\hat{y}_2 = 2$
$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}]$	$y_3 = 1$	$\hat{y}_3 = 2$
•		
•		
•		
$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}]$	$y_n = 3$	$\hat{y}_n = 1$

We need to find a function that maps x and y for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we "learn" the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^n Cost(\hat{y}_i, y_i)$$

Supervised Learning - Classification

Training Data



cat



dog



cat

.

.

.



bear

Test Data



.

.

.



Supervised Learning - Classification

Training Data

$$x_1 = [\text{] \quad y_1 = [\text{cat}]$$

$$x_2 = [\text{] \quad y_2 = [\text{dog}]$$

$$x_3 = [\text{] \quad y_3 = [\text{cat}]$$

•
•
•

$$x_n = [\text{] \quad y_n = [\text{bear}]$$

Supervised Learning - Classification

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}]$	$y_1 = 1$	$\hat{y}_1 = 1$
$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}]$	$y_2 = 2$	$\hat{y}_2 = 2$
$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}]$	$y_3 = 1$	$\hat{y}_3 = 2$
•		
•		
•		
$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}]$	$y_n = 3$	$\hat{y}_n = 1$

We need to find a function that maps x and y for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we "learn" the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^n Cost(\hat{y}_i, y_i)$$

Supervised Learning – Linear Softmax

Training Data

inputs

targets /
labels /
ground truth

$$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}] \quad y_1 = 1$$

$$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}] \quad y_2 = 2$$

$$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}] \quad y_3 = 1$$

•
•
•

$$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}] \quad y_n = 3$$

Supervised Learning – Linear Softmax

Training Data

inputs	targets / labels / ground truth	predictions
$x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}]$	$y_1 = [1 \ 0 \ 0]$	$\hat{y}_1 = [0.85 \ 0.10 \ 0.05]$
$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}]$	$y_2 = [0 \ 1 \ 0]$	$\hat{y}_2 = [0.20 \ 0.70 \ 0.10]$
$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}]$	$y_3 = [1 \ 0 \ 0]$	$\hat{y}_3 = [0.40 \ 0.45 \ 0.15]$
•		
•		
•		
$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}]$	$y_n = [0 \ 0 \ 1]$	$\hat{y}_n = [0.40 \ 0.25 \ 0.35]$

Supervised Learning – Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_1 \ f_2 \ f_3]$$

$$a_1 = w_{11}x_{i1} + w_{12}x_{i2} + w_{13}x_{i3} + w_{14}x_{i4} + b_c$$

$$a_2 = w_{21}x_{i1} + w_{22}x_{i2} + w_{23}x_{i3} + w_{24}x_{i4} + b_d$$

$$a_3 = w_{31}x_{i1} + w_{32}x_{i2} + w_{33}x_{i3} + w_{34}x_{i4} + b_b$$

$$f_1 = e^{a_1} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_2 = e^{a_2} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_3 = e^{a_3} / (e^{a_1} + e^{a_2} + e^{a_3})$$

How do we find a good w and b?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_1(w, b) \ f_2(w, b) \ f_3(w, b)]$$

We need to find w, and b that minimize the following:

$$L(w, b) = \sum_{i=1}^n \sum_{j=1}^3 -y_{i,j} \log(\hat{y}_{i,j}) = \sum_{i=1}^n -\log(\hat{y}_{i, label}) = \sum_{i=1}^n -\log f_{i, label}(w, b)$$

Why?

(mini-batch) Stochastic Gradient Descent (SGD)

$\lambda = 0.01$

Initialize w and b randomly

$$l(w, b) = \sum_{i \in B} \text{Cost}(w, b)$$

for $e = 0, \text{num_epochs}$ **do**

for $b = 0, \text{num_batches}$ **do**

 Compute: $dl(w, b)/dw$ and $dl(w, b)/db$

 Update w : $w = w - \lambda dl(w, b)/dw$

 Update b : $b = b - \lambda dl(w, b)/db$

 Print: $l(w, b)$ *// Useful to see if this is becoming smaller or not.*

end

end

Computing Analytic Gradients

This is what we have:

$$L(w, b) = \sum_{i=1}^n \sum_{j=1}^3 -y_{i,j} \log(\hat{y}_{i,j}) = \sum_{i=1}^n -\log(\hat{y}_{i,label}) = \sum_{i=1}^n -\log f_{i,label}(w, b)$$

To simplify let's assume $n = 1$

$$\ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left(\frac{\exp(a_{label}(W, b))}{\sum_{k=1}^3 \exp(a_k(W, b))}\right)$$

Supervised Learning – Linear Softmax

$$x = [x_1 \ x_2 \ x_3 \ x_4] \qquad y = [1 \ 0 \ 0] \qquad \hat{y} = [f_1 \ f_2 \ f_3]$$

$$a_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + b_c$$

$$a_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + b_d$$

$$a_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 + b_b$$

$$f_1 = e^{a_1} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_2 = e^{a_2} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_3 = e^{a_3} / (e^{a_1} + e^{a_2} + e^{a_3})$$

Computing Analytic Gradients

This is what we have:

$$\ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left(\frac{\exp(a_{label}(W, b))}{\sum_{k=1}^3 \exp(a_k(W, b))}\right)$$

Computing Analytic Gradients

This is what we have:

$$\ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left(\frac{\exp(a_{label}(W, b))}{\sum_{k=1}^3 \exp(a_k(W, b))}\right)$$

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Reminder: $a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

This is what we need:

$$\frac{\partial \ell}{\partial w_{ij}} \quad \text{for each } w_{ij}$$

$$\frac{\partial \ell}{\partial b_i} \quad \text{for each } b_i$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i}$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

Let's do these first

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \boxed{\frac{\partial a_i}{\partial w_{ij}}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \boxed{\frac{\partial a_i}{\partial b_i}}$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial a_i}{\partial b_i}$$

$$a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial w_{i,3}} = \frac{\partial}{\partial w_{i,3}} (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial w_{i,3}} = x_3$$

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

$$\frac{\partial a_i}{\partial b_i}$$

$$a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial b_i} = \frac{\partial}{\partial b_i} (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i$$

$$\frac{\partial a_i}{\partial b_i} = 1$$

Computing Analytic Gradients

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

$$\frac{\partial a_i}{\partial b_i} = 1$$

Computing Analytic Gradients

This is what we have:

$$\ell = -\log\left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)}\right)$$

Step 1: Chain Rule of Calculus

Now let's do this one (same for both!)

$$\frac{\partial \ell}{\partial w_{ij}} = \boxed{\frac{\partial \ell}{\partial a_i}} \frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial \ell}{\partial b_i} = \boxed{\frac{\partial \ell}{\partial a_i}} \frac{\partial a_i}{\partial b_i}$$

Computing Analytic Gradients

$$\begin{aligned}\frac{\partial \ell}{\partial a_i} &= \frac{\partial}{\partial a_i} \left[-\log \left(\frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)} \right) \right] \\ &= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]\end{aligned}$$

In our cat, dog, bear classification example: $i = \{1, 2, 3\}$

Computing Analytic Gradients

$$\begin{aligned}\frac{\partial \ell}{\partial a_i} &= \frac{\partial}{\partial a_i} \left[-\log \left(\frac{\exp(a_{\text{label}})}{\sum_{k=1}^3 \exp(a_k)} \right) \right] \\ &= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{\text{label}} \right]\end{aligned}$$

In our cat, dog, bear classification example: $i = \{1, 2, 3\}$

Let's say: label = 2

We need:

$$\frac{\partial \ell}{\partial a_1} \quad \frac{\partial \ell}{\partial a_2} \quad \frac{\partial \ell}{\partial a_3}$$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{\text{label}} \right]$$

$$\frac{\partial \ell}{\partial a_1} \quad \frac{\partial \ell}{\partial a_3} \quad \text{when } i \neq \text{label:}$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{\text{label}} \right]$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \log \left(\sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \left(\frac{1}{\sum_{k=1}^3 \exp(a_k)} \right) \left(\frac{\partial}{\partial a_i} \sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\exp(a_i)}{\sum_{k=1}^3 \exp(a_k)} = \hat{y}_i$$

Supervised Learning – Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_1 \ f_2 \ f_3]$$

$$a_1 = w_{11}x_{i1} + w_{12}x_{i2} + w_{13}x_{i3} + w_{14}x_{i4} + b_c$$

$$a_2 = w_{21}x_{i1} + w_{22}x_{i2} + w_{23}x_{i3} + w_{24}x_{i4} + b_d$$

$$a_3 = w_{31}x_{i1} + w_{32}x_{i2} + w_{33}x_{i3} + w_{34}x_{i4} + b_b$$

$$f_1 = e^{a_1} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_2 = e^{a_2} / (e^{a_1} + e^{a_2} + e^{a_3})$$

$$f_3 = e^{a_3} / (e^{a_1} + e^{a_2} + e^{a_3})$$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_1} \quad \frac{\partial \ell}{\partial a_3} \quad \text{when } i \neq label:$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \log \left(\sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \left(\frac{1}{\sum_{k=1}^3 \exp(a_k)} \right) \left(\frac{\partial}{\partial a_i} \sum_{k=1}^3 \exp(a_k) \right)$$

$$\frac{\partial \ell}{\partial a_i} = \frac{\exp(a_i)}{\sum_{k=1}^3 \exp(a_k)} = \hat{y}_i$$

Computing Analytic Gradients

$$= \frac{\partial}{\partial a_i} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_2}$$

when $i = label$:

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\partial}{\partial a_{label}} \left[\log \left(\sum_{k=1}^3 \exp(a_k) \right) - a_{label} \right]$$

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\partial}{\partial a_{label}} \log \left(\sum_{k=1}^3 \exp(a_k) \right) - 1$$

$$\frac{\partial \ell}{\partial a_{label}} = \left(\frac{1}{\sum_{k=1}^3 \exp(a_k)} \right) \left(\frac{\partial}{\partial a_{label}} \sum_{k=1}^3 \exp(a_k) \right) - 1$$

$$\frac{\partial \ell}{\partial a_{label}} = \frac{\exp(a_{label})}{\sum_{k=1}^3 \exp(a_k)} - 1 \quad \hat{y}_i - 1$$

Computing Analytic Gradients

label = 2

$$\frac{\partial \ell}{\partial a_1} = \hat{y}_1$$

$$\frac{\partial \ell}{\partial a_2} = \hat{y}_2 - 1$$

$$\frac{\partial \ell}{\partial a_3} = \hat{y}_3$$

$$\frac{\partial \ell}{\partial a} = \begin{bmatrix} \frac{\partial \ell}{\partial a_1} \\ \frac{\partial \ell}{\partial a_2} \\ \frac{\partial \ell}{\partial a_3} \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 - 1 \\ \hat{y}_3 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \hat{y} - y$$

$$\frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i$$

Computing Analytic Gradients

$$\frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}}$$

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i}$$

$$\frac{\partial a_i}{\partial w_{i,j}} = x_j$$

$$\frac{\partial a_i}{\partial b_i} = 1$$

$$\frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i$$

$$\frac{\partial \ell}{\partial w_{i,j}} = (\hat{y}_i - y_i)x_j$$

$$\frac{\partial \ell}{\partial b_i} = (\hat{y}_i - y_i)$$

Automatic Differentiation

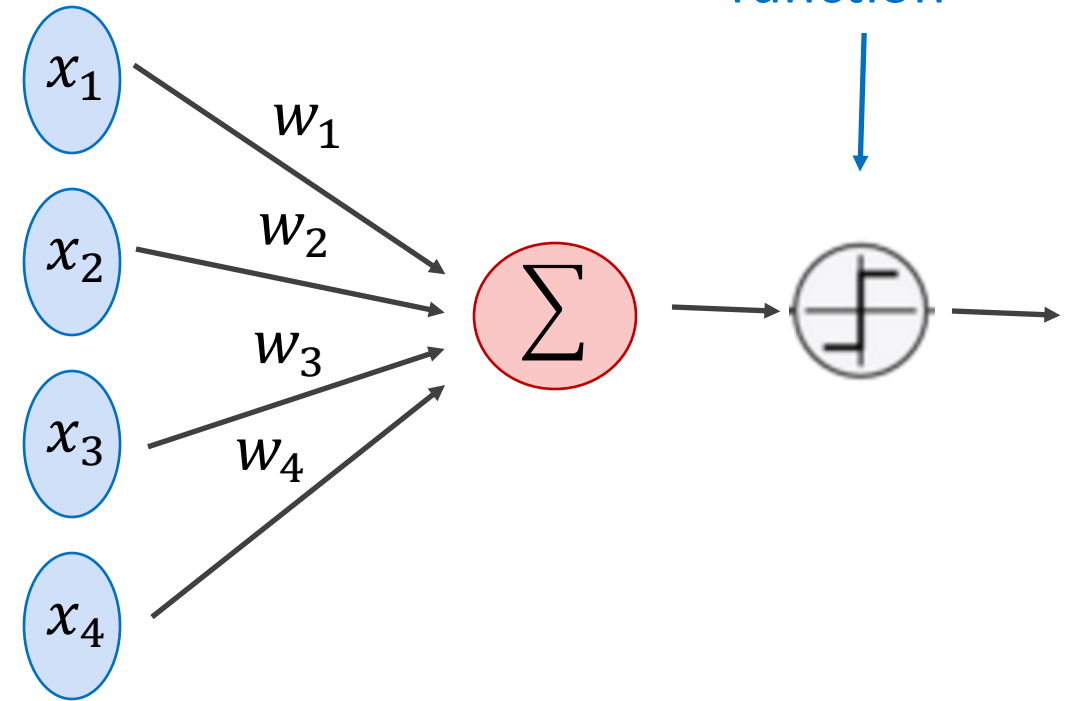
You only need to write code for the operations in the prediction step,
Gradient computation can be computed “automatically”.

Pytorch (Facebook -- mostly):	https://pytorch.org/
Tensorflow (Google -- mostly):	https://www.tensorflow.org/
MXNet (Amazon -- mostly):	https://mxnet.apache.org/versions/1.9.0/

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^n w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

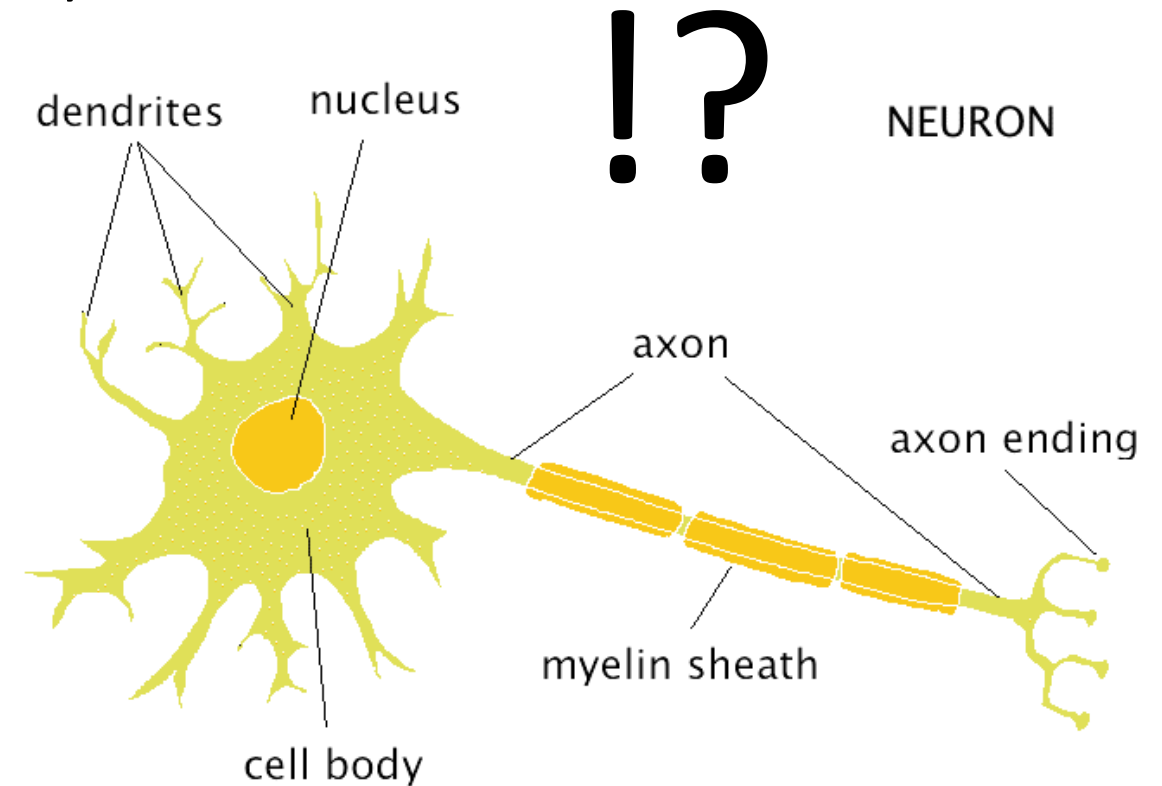


More: <https://en.wikipedia.org/wiki/Perceptron>

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^n w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

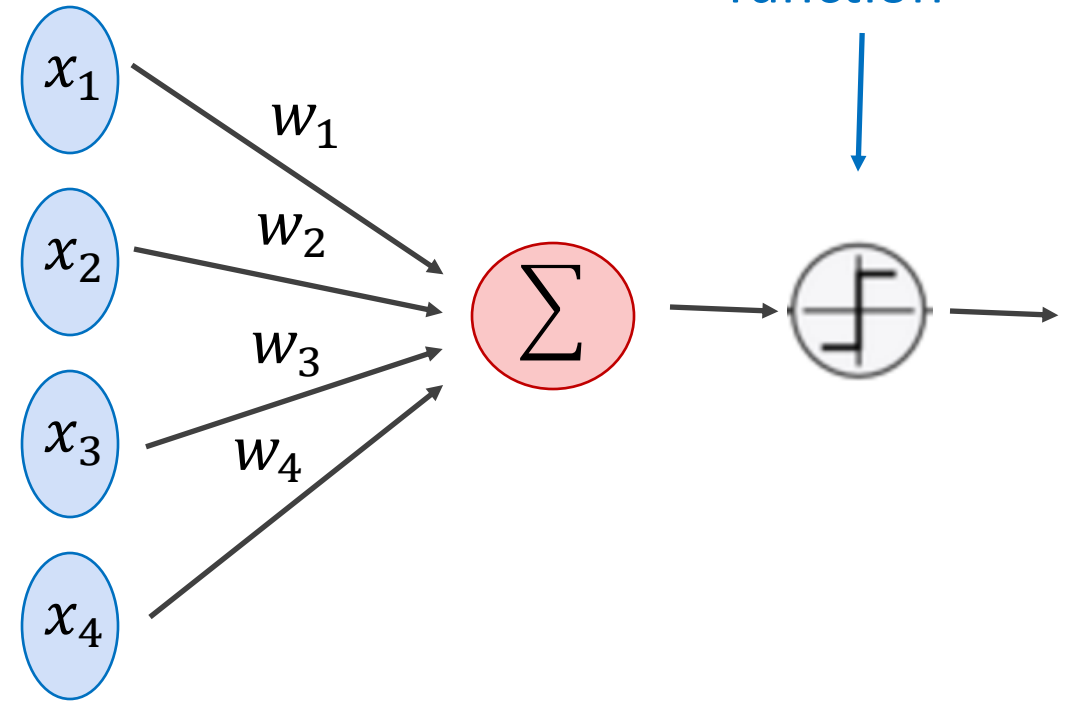


More: <https://en.wikipedia.org/wiki/Perceptron>

Perceptron Model

Frank Rosenblatt (1957) - Cornell University

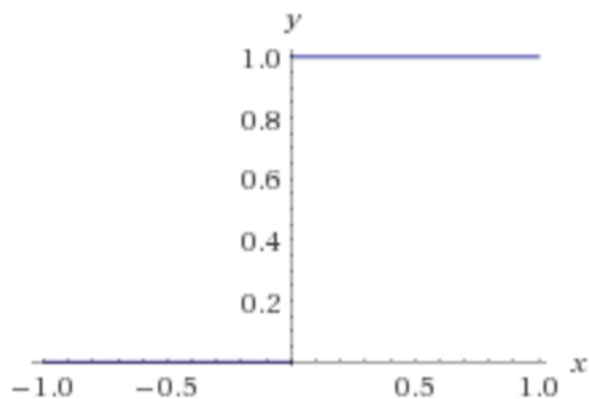
$$f(x) = \begin{cases} 1, & \text{if } \sum_{i=0}^n w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases}$$



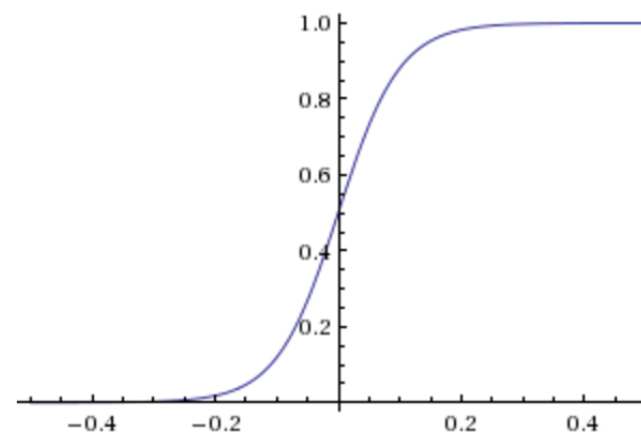
More: <https://en.wikipedia.org/wiki/Perceptron>

Activation Functions

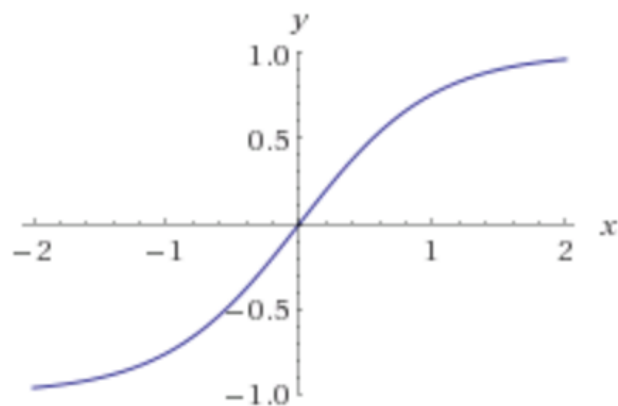
Step(x)



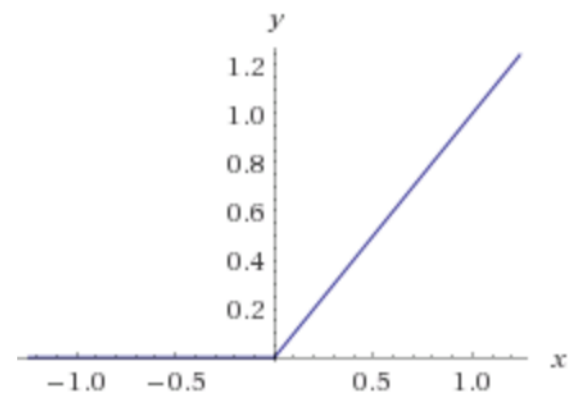
Sigmoid(x)



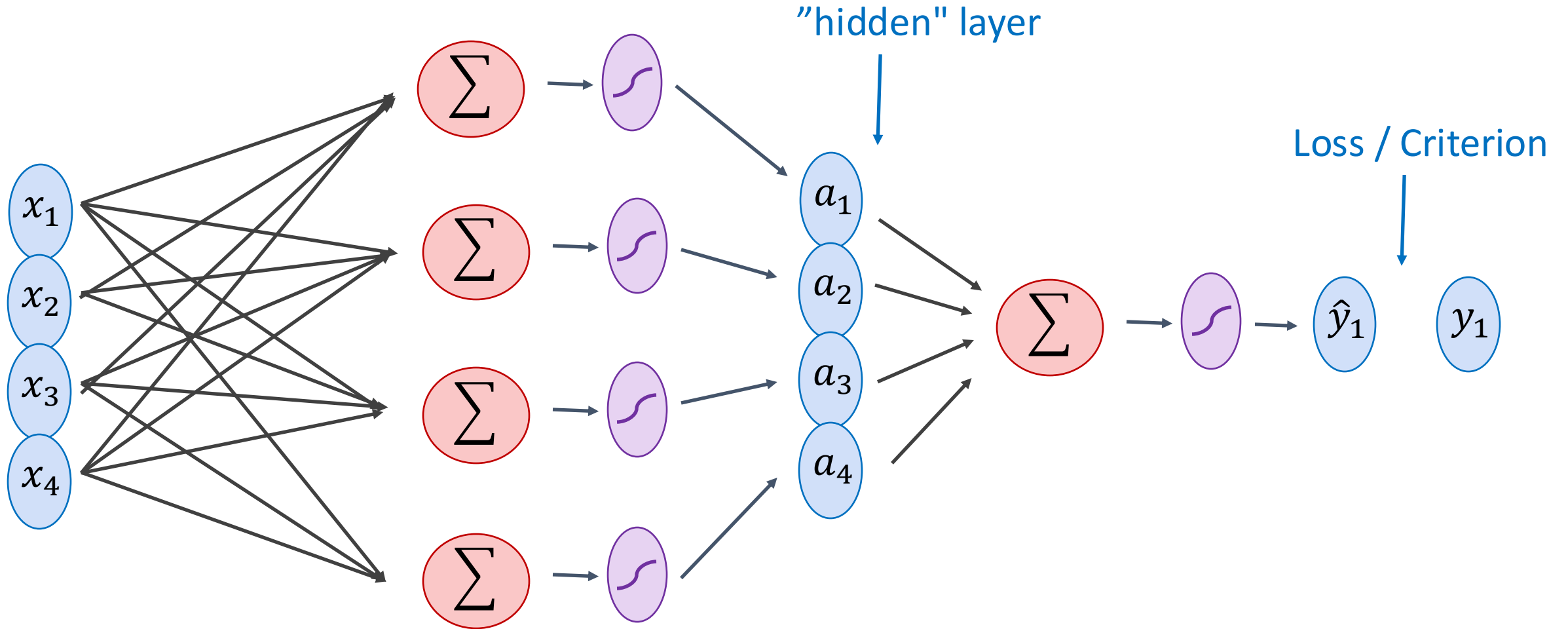
Tanh(x)



$\text{ReLU}(x) = \max(0, x)$



Two-layer Multi-layer Perceptron (MLP)



Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$

$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$

$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b} / (e^{g_c} + e^{g_d} + e^{g_b})$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$

$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$

$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$W = \begin{bmatrix} w_{c1} & w_{c2} & w_{c3} & w_{c4} \\ w_{d1} & w_{d2} & w_{d3} & w_{d4} \\ w_{b1} & w_{b2} & w_{b3} & w_{b4} \end{bmatrix}$$

$$b = [b_c \ b_d \ b_b]$$

$$f_c = e^{g_c} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b} / (e^{g_c} + e^{g_d} + e^{g_b})$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g = wx^T + b^T$$

$$w = \begin{bmatrix} w_{c1} & w_{c2} & w_{c3} & w_{c4} \\ w_{d1} & w_{d2} & w_{d3} & w_{d4} \\ w_{b1} & w_{b2} & w_{b3} & w_{b4} \end{bmatrix}$$

$$b = [b_c \ b_d \ b_b]$$

$$f_c = e^{g_c} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_d = e^{g_d} / (e^{g_c} + e^{g_d} + e^{g_b})$$

$$f_b = e^{g_b} / (e^{g_c} + e^{g_d} + e^{g_b})$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_d \ f_b]$$

$$g = wx^T + b^T$$

$$w = \begin{bmatrix} w_{c1} & w_{c2} & w_{c3} & w_{c4} \\ w_{d1} & w_{d2} & w_{d3} & w_{d4} \\ w_{b1} & w_{b2} & w_{b3} & w_{b4} \end{bmatrix}$$

$$b = [b_c \ b_d \ b_b]$$

$$f = \text{softmax}(g)$$

Linear Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$f = \text{softmax}(wx^T + b^T)$$

Two-layer MLP + Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$f = \text{softmax}(w_{[2]}a_1^T + b_{[2]}^T)$$

N-layer MLP + Softmax

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_k = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

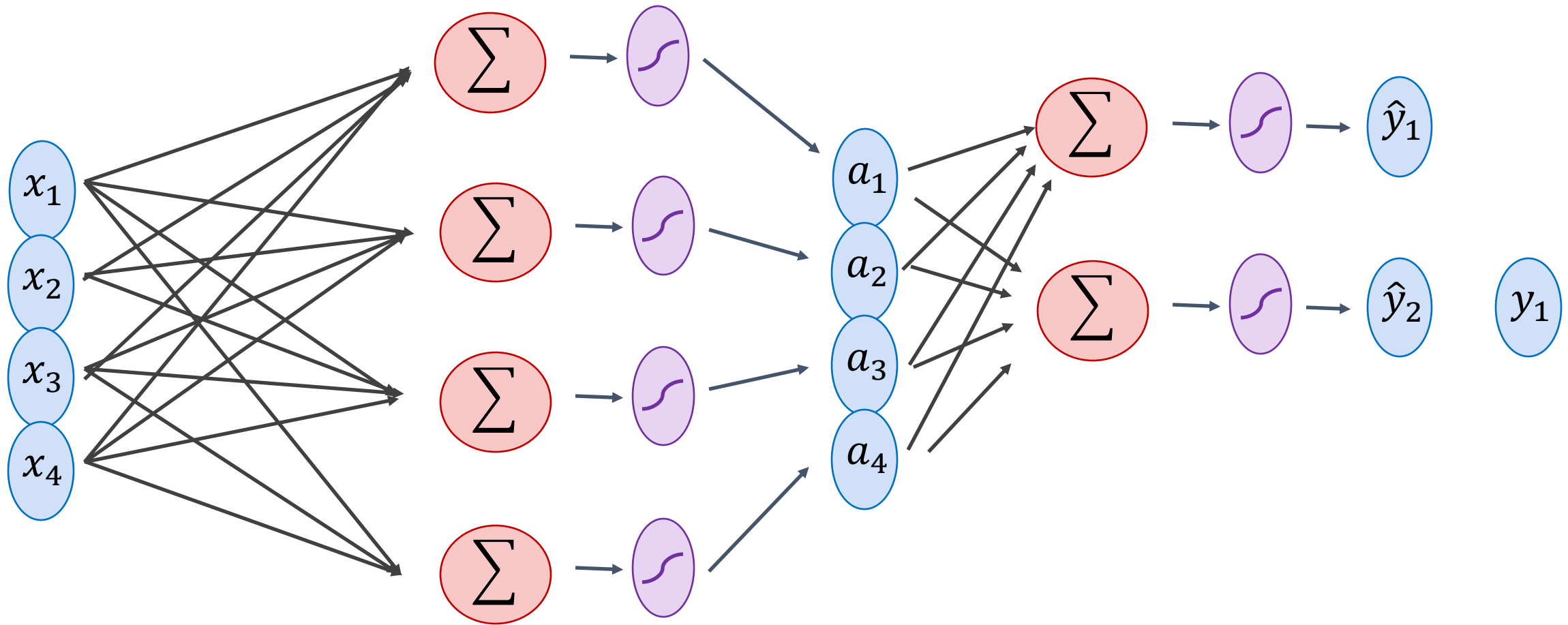
...

$$a_k = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

Forward pass (Forward-propagation)



Forward pass (Forward-propagation)

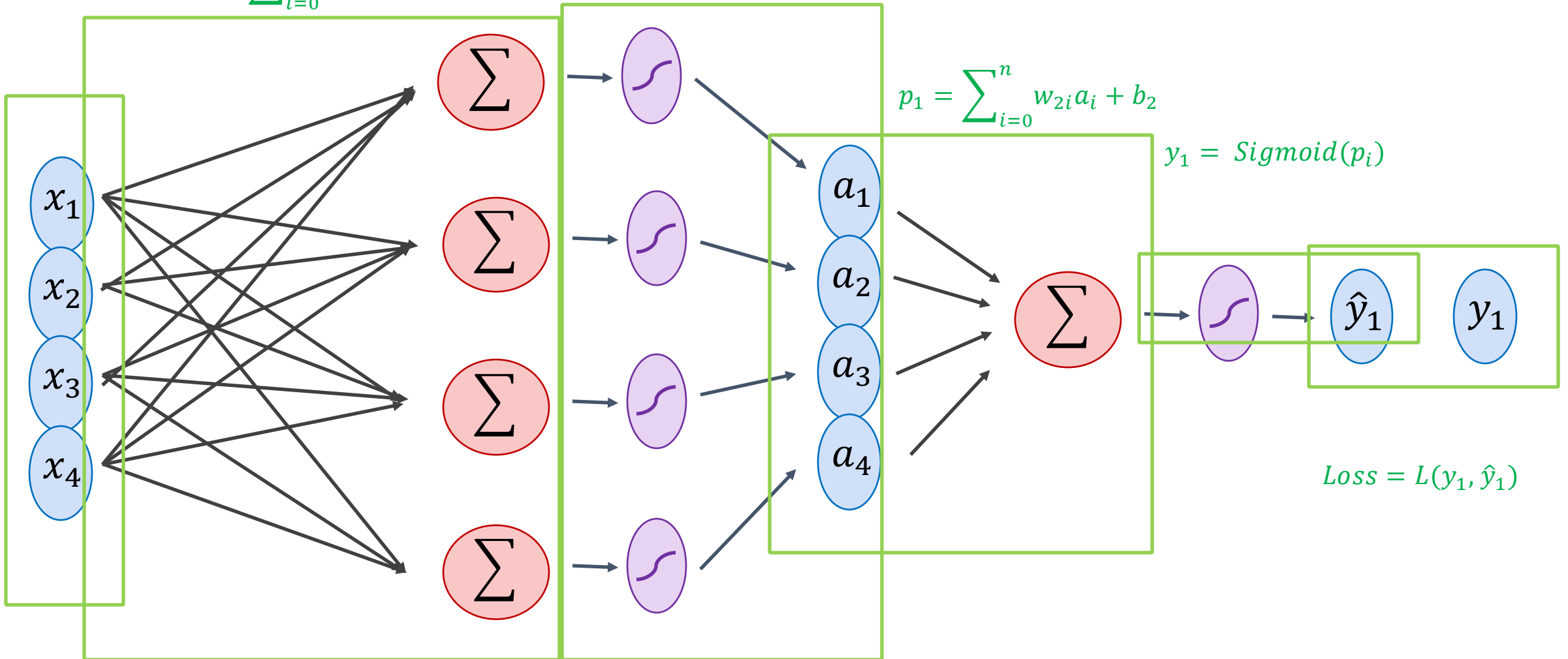
$$z_i = \sum_{i=0}^n w_{1ij}x_i + b_1$$

$$a_i = \text{Sigmoid}(z_i)$$

$$p_1 = \sum_{i=0}^n w_{2i}a_i + b_2$$

$$y_1 = \text{Sigmoid}(p_1)$$

$$\text{Loss} = L(y_1, \hat{y}_1)$$



How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_k = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

We can still use SGD

We need!

$$\frac{\partial l}{\partial w_{[k]ij}}$$

$$\frac{\partial l}{\partial b_{[k]i}}$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_i = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

$$l = \text{loss}(f, y)$$

We can still use SGD

We need!

$$\frac{\partial l}{\partial w_{[k]ij}}$$

$$\frac{\partial l}{\partial b_{[k]i}}$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_i = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

$$l = \text{loss}(f, y)$$

We can still use SGD

We need!

$$\frac{\partial l}{\partial w_{[k]ij}}$$

$$\frac{\partial l}{\partial b_{[k]i}}$$

How to train the parameters?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]$$

$$y_i = [1 \ 0 \ 0]$$

$$\hat{y}_i = [f_c \ f_a \ f_b]$$

$$a_1 = \text{sigmoid}(w_{[1]}x^T + b_{[1]}^T)$$

$$a_2 = \text{sigmoid}(w_{[2]}a_1^T + b_{[2]}^T)$$

...

$$a_i = \text{sigmoid}(w_{[k]}a_{k-1}^T + b_{[k]}^T)$$

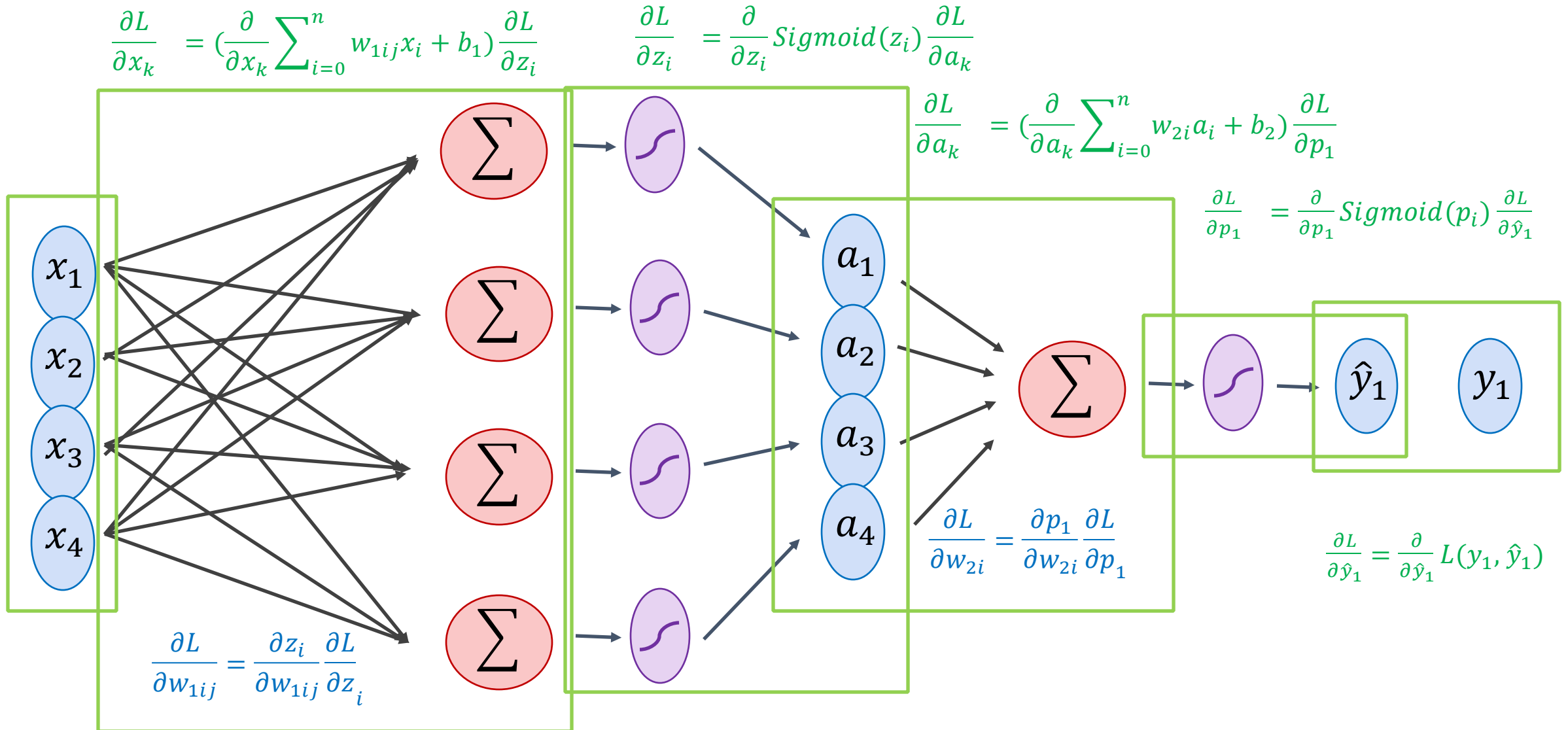
...

$$f = \text{softmax}(w_{[n]}a_{n-1}^T + b_{[n]}^T)$$

$$l = \text{loss}(f, y)$$

$$\frac{\partial l}{\partial w_{[k]ij}} = \frac{\partial l}{\partial a_{n-1}} \frac{\partial a_{n-1}}{\partial a_{n-2}} \cdots \frac{\partial a_{k-2}}{\partial a_{k-1}} \frac{\partial a_{k-1}}{\partial w_{[k]ij}}$$

Backward pass (Back-propagation)



Automatic Differentiation

You only need to write code for the forward pass,
backward pass is computed automatically.

Pytorch (Facebook -- mostly):	https://pytorch.org/
Tensorflow (Google -- mostly):	https://www.tensorflow.org/
MXNet (Amazon -- mostly):	https://mxnet.apache.org/versions/1.9.0/

Defining a Model in Pytorch (Two Layer NN)

```
import torch.nn as nn
import torch.nn.functional as F

class TwoLayerNN(nn.Module):
    def __init__(self):
        super(TwoLayerNN, self).__init__()

        self.linear1 = nn.Linear(1 * 28 * 28, 512)
        self.linear2 = nn.Linear(512, 10)

    def forward(self, x):
        x = x.view(batchSize, 1 * 28 * 28)
        z = F.relu(self.linear1(x))
        return self.linear2(z)
```


2. Running forward and backward on a batch

```
# Forward pass. (Prediction stage)
scores = model(inputs)
loss = loss_fn(scores, labels)
```

```
# Zero the gradients in the network.
optimizer.zero_grad()

#Backward pass. (Gradient computation stage)
loss.backward()

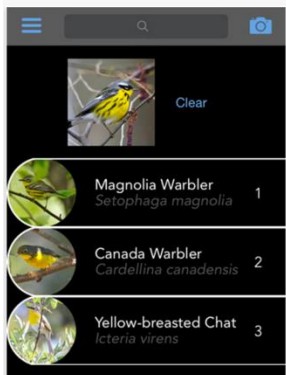
# Parameter updates (SGD step) -- if done with torch.optim!
optimizer.step()
```



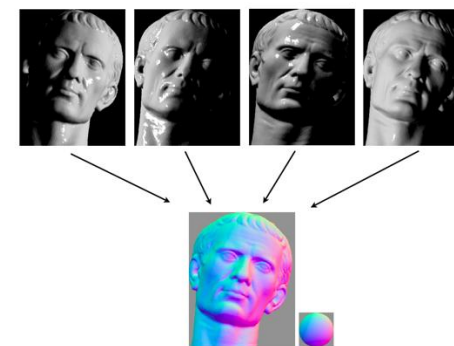
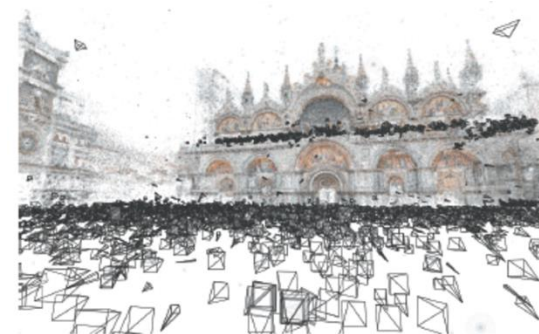
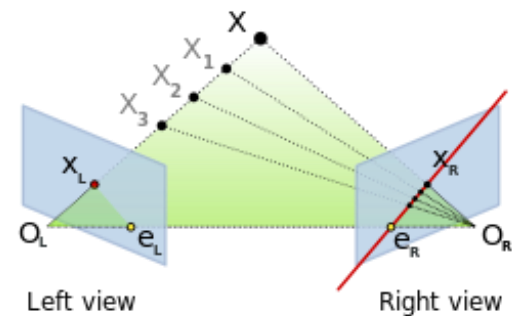
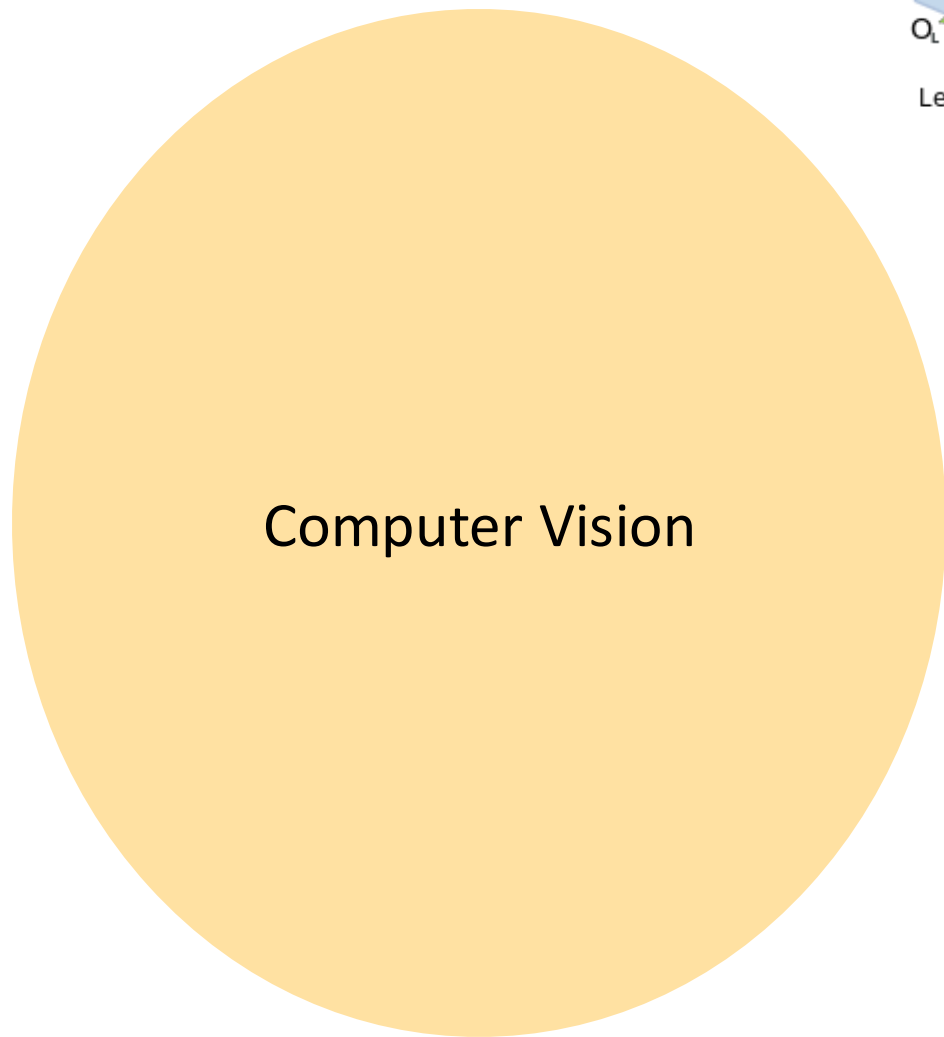
Create an algorithm to distinguish dogs from cats

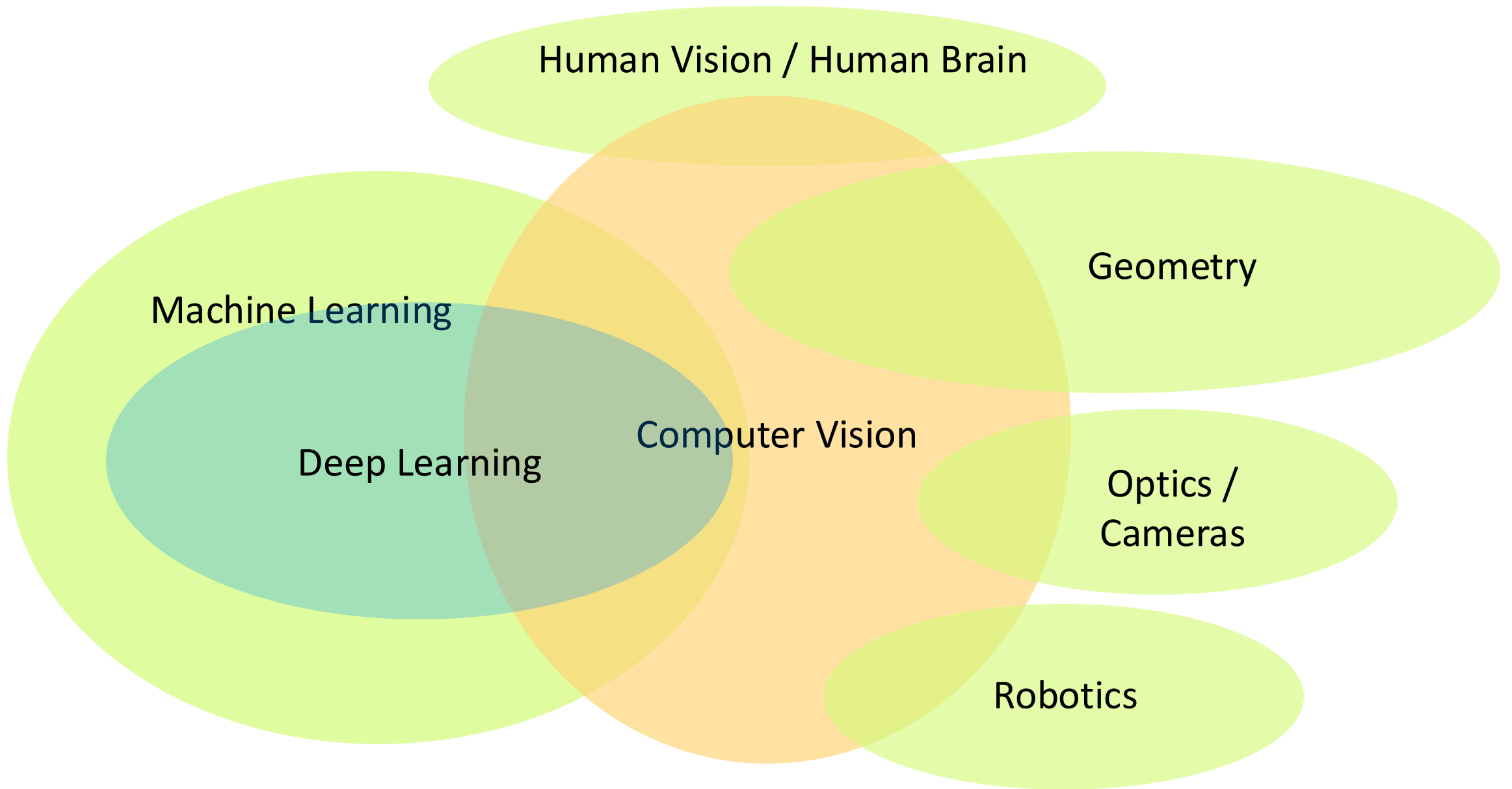


Birdsnap



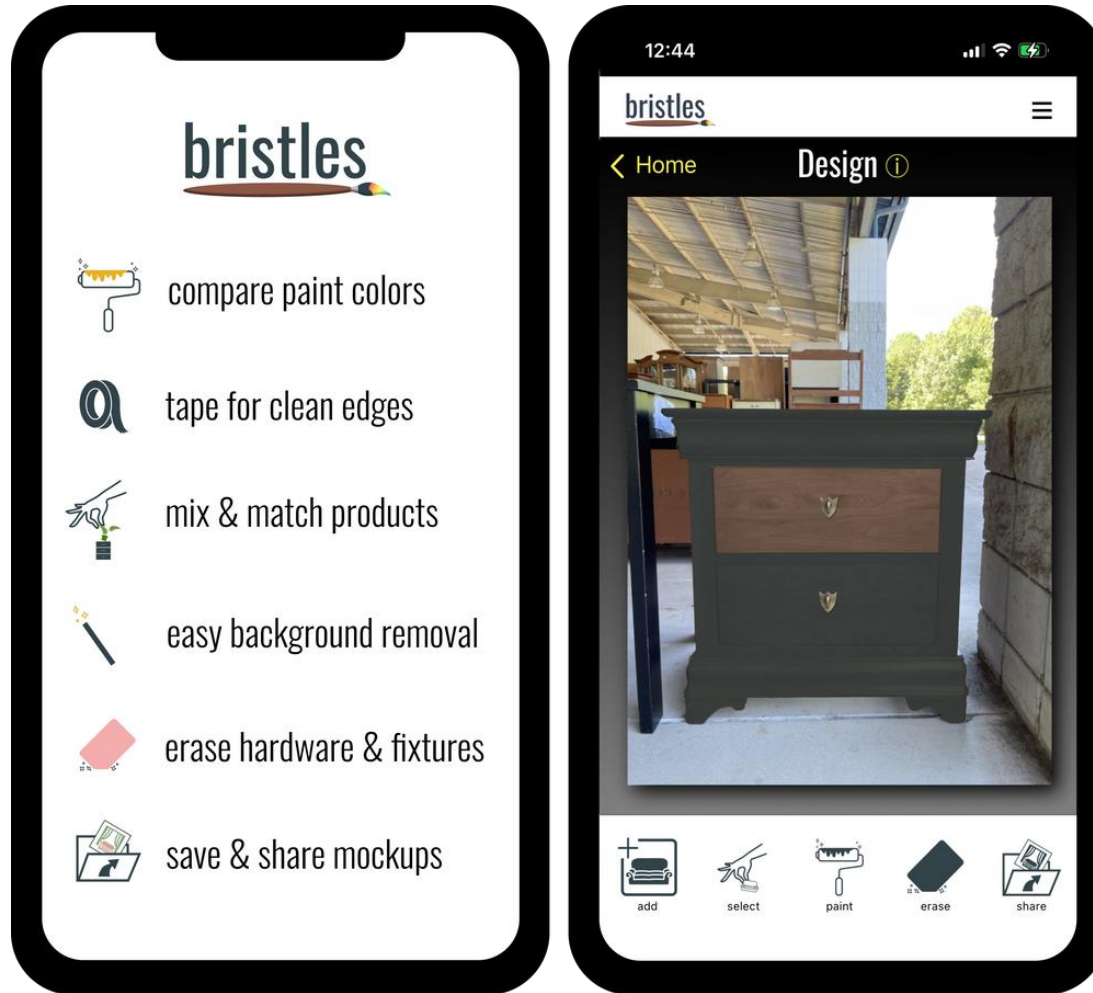
Face Detection in Cameras



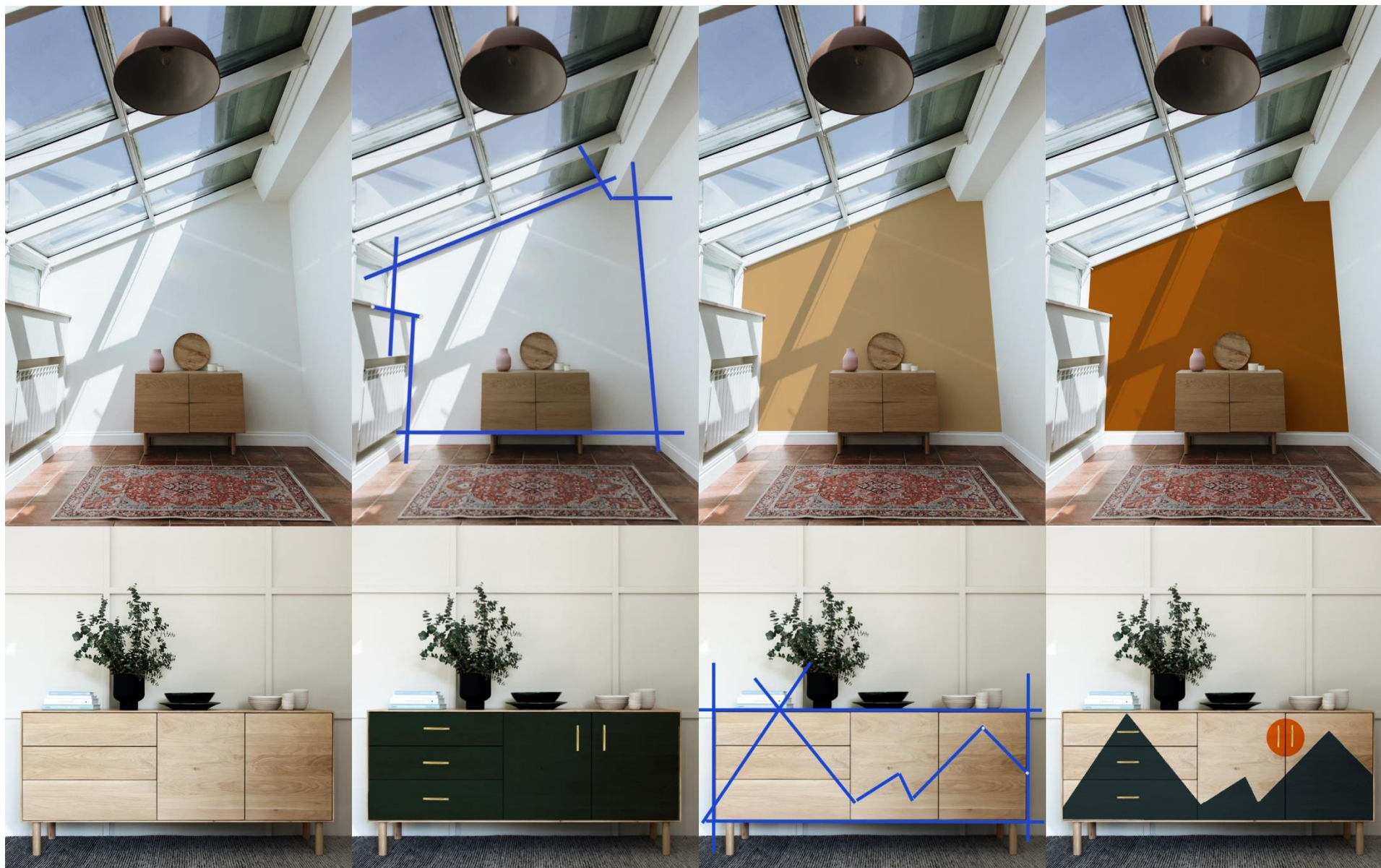


Who is using Computer Vision?

- Facebook – Oculus VR, Image Search, Image tagging, Content filtering, Instagram, etc.
- Google/Alphabet – Waymo, DeepMind, Image Search, Google Earth/Maps, Street View, Google Photos, etc.
- Adobe – Photoshop, Premiere, Lightroom, etc.
- Snap Inc – Snapchat, Smart Goggles, Filters, Face Detection, Style Transfer, etc.
- eBay Inc – Product Search, Product Matching, Content Filtering, Duplicate Removal, etc.
- Amazon – Warehouse robotics, Smart Stores, Product Search.
- IBM – Image Retrieval, Medical Applications, Product Quality.
- Microsoft – Hololens, Optical Character Recognition (OCR), Face Detection, Cloud Services.
- Apple – Face Verification, Enhanced cameras and chips for image processing.



<https://bristles.ai/>



<https://bristles.ai/>

Health & Safety for Loved Ones, Peace-of-Mind for Caregivers

Remote AI older adult monitoring at home and in communities

Order Now

▶ Product Demo

Current Status



Mom
Fallen Out of Bed for 2 minutes

Safety Score

55

Your nightly average: 60

Last Nights Detection





Phiar.ai (now part of Google)

Images

- Can be viewed as a matrix with pixel values

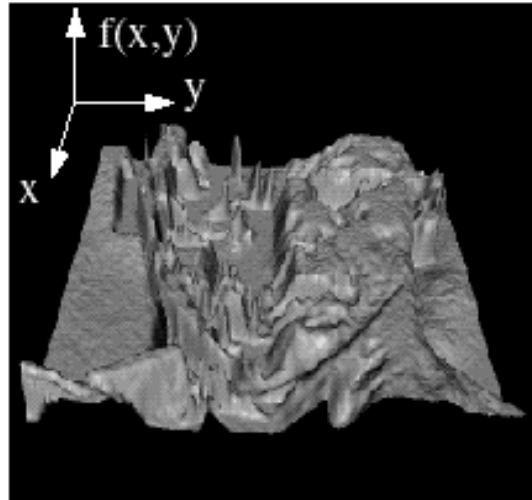


0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

Images

- Or as a function in a 2D domain

$$z = f(x, y)$$



Color Images

- Can be viewed as tensors (3-dimensional arrays)



$T =$

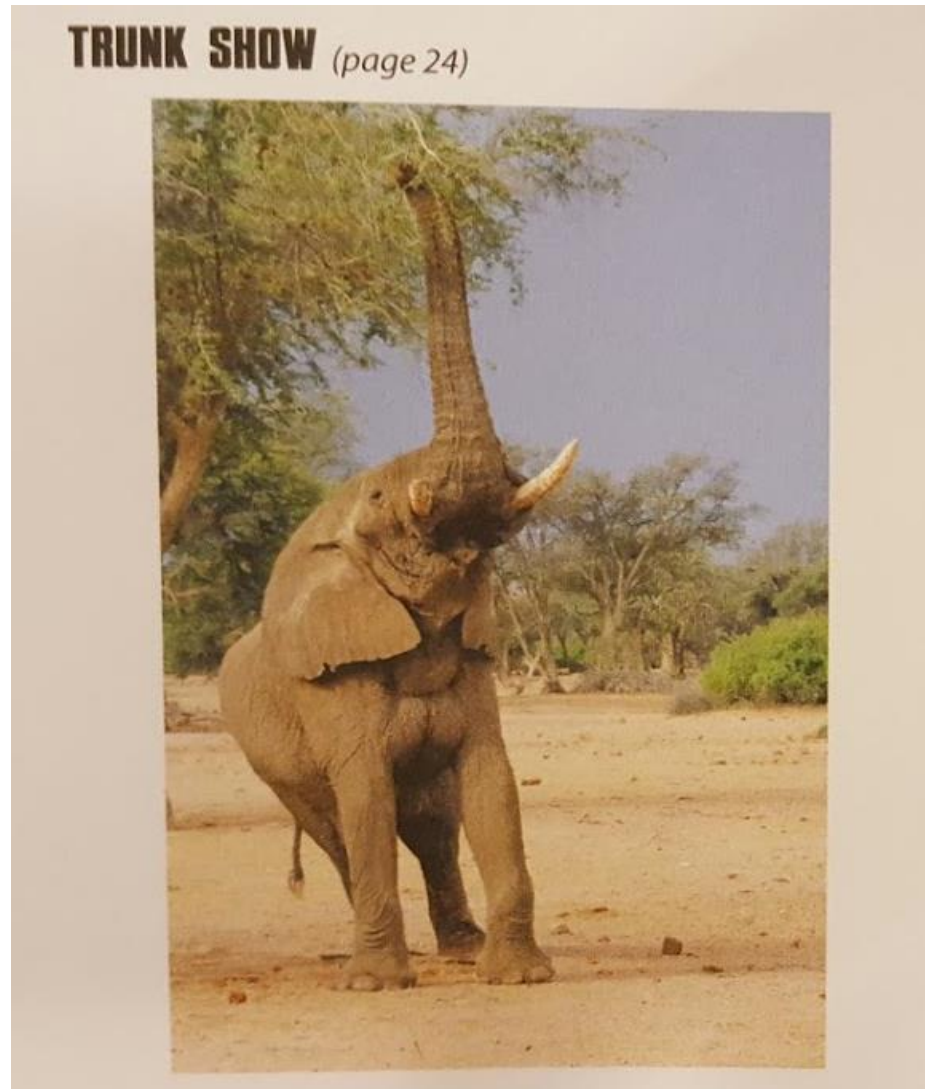
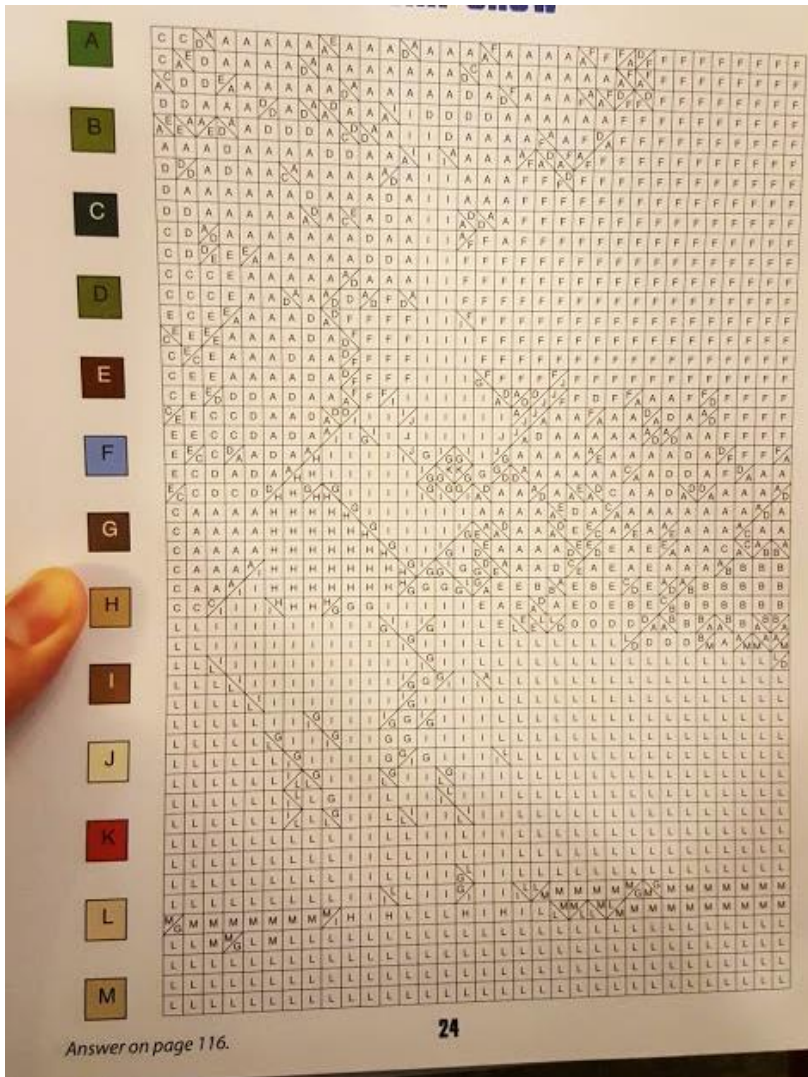
0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

$\text{sizeof}(T) = 3 \times \text{height} \times \text{width}$

Channels are usually RGB: Red, Green, and Blue

Other color spaces: HSV, HSL, LUV, XYZ, Lab, CMYK, etc

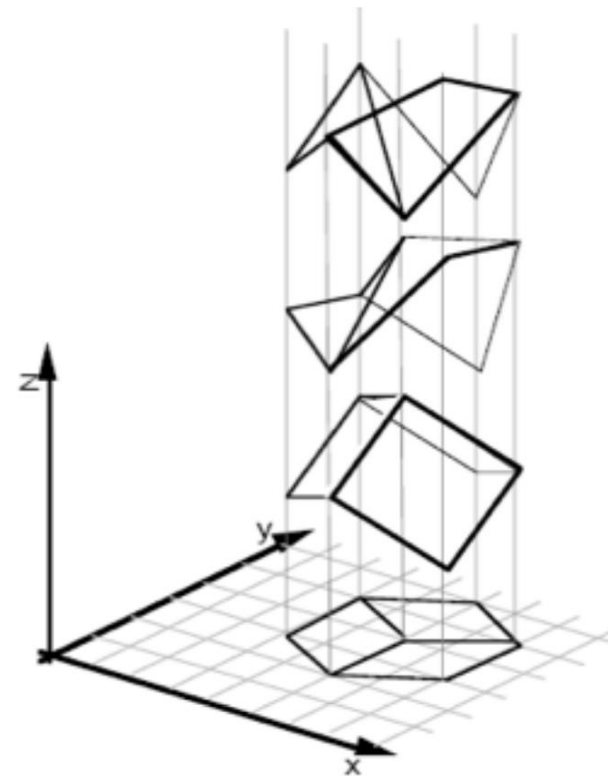
Why is it hard?



This is just as hard for computers

Why is Computer Vision hard?

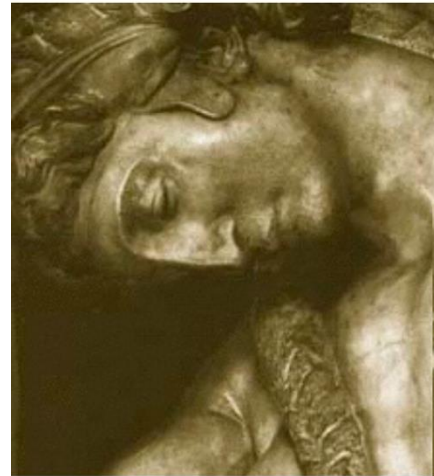
Ambiguities due to viewpoints



[Sinha and Adelson 1993]

Why is Computer Vision hard?

Ambiguities due to viewpoints



Michelangelo 1475-1564



slide by Fei Fei, Fergus & Torralba

Why is Computer Vision hard?

Issues with
Illumination



slide credit: S. Ullman

Why is Computer Vision hard?

Background
clutter



Why is Computer Vision hard?

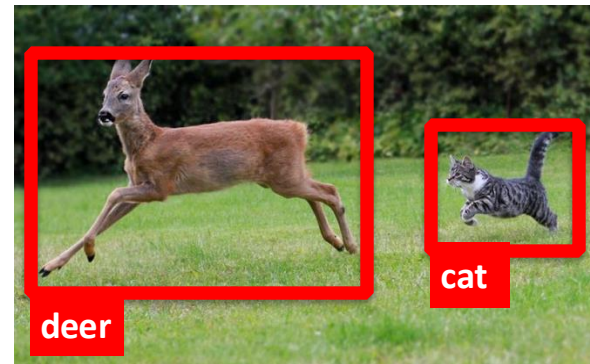
Intra-class
variation



slide by Fei-Fei, Fergus & Torralba

Computer Vision vs Image Processing

- Computer Vision: Image $\longrightarrow$ Knowledge



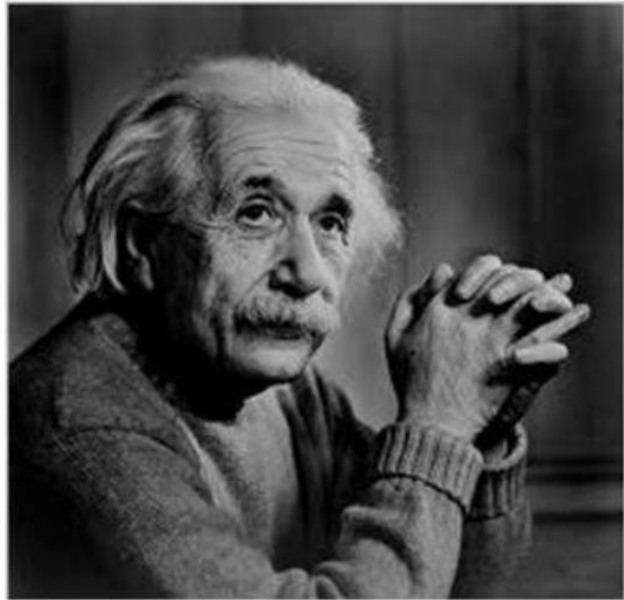
Computer Vision vs Image Processing

- Image Processing: Image $\longrightarrow$ Image

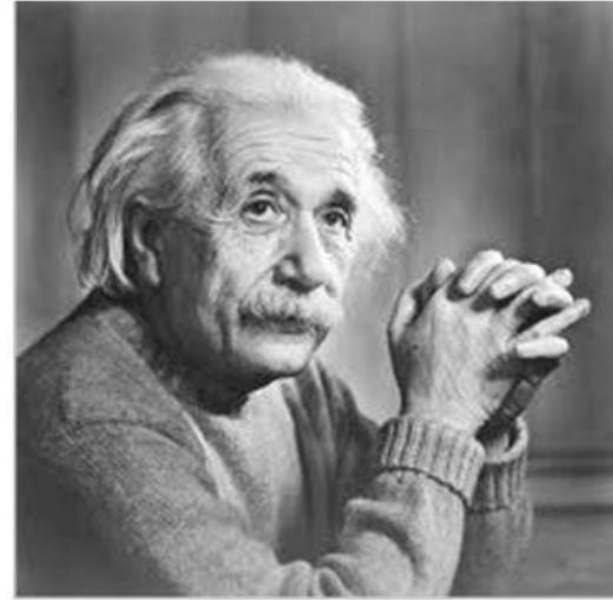


Basic Image Processing

I



αI

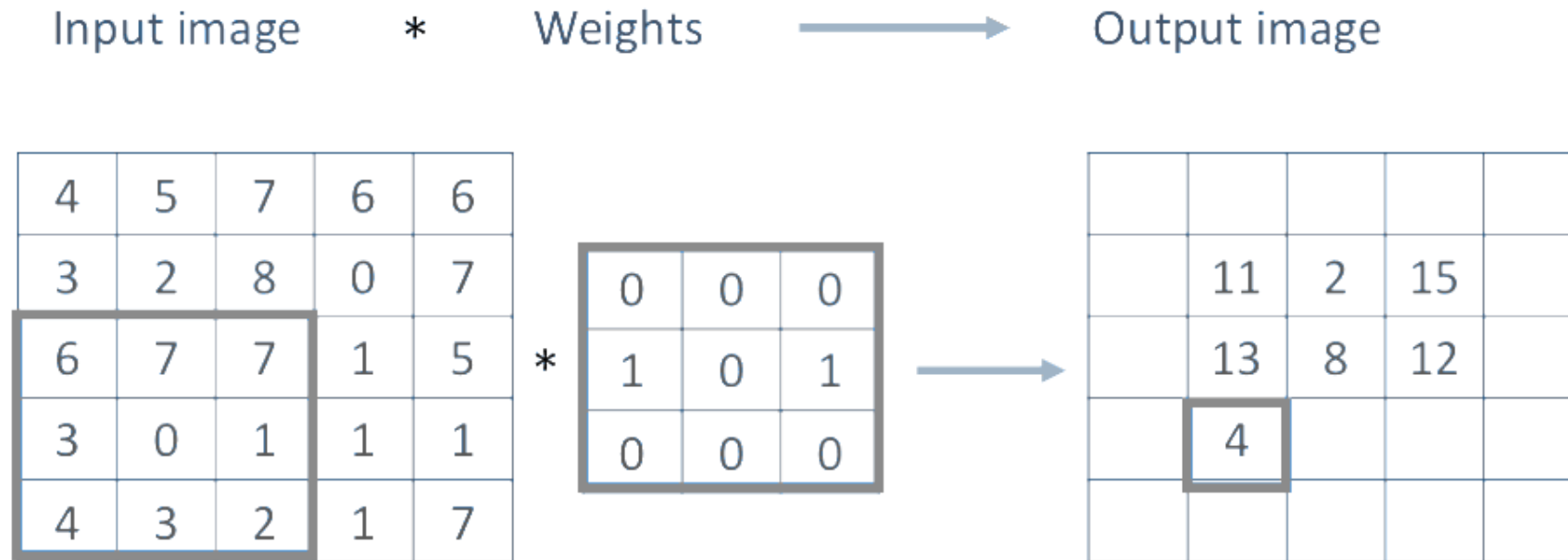


$\alpha > 1$

Primer on Image Processing: <https://bit.ly/3lGEwv>

Most important operation for Computer Vision (*)

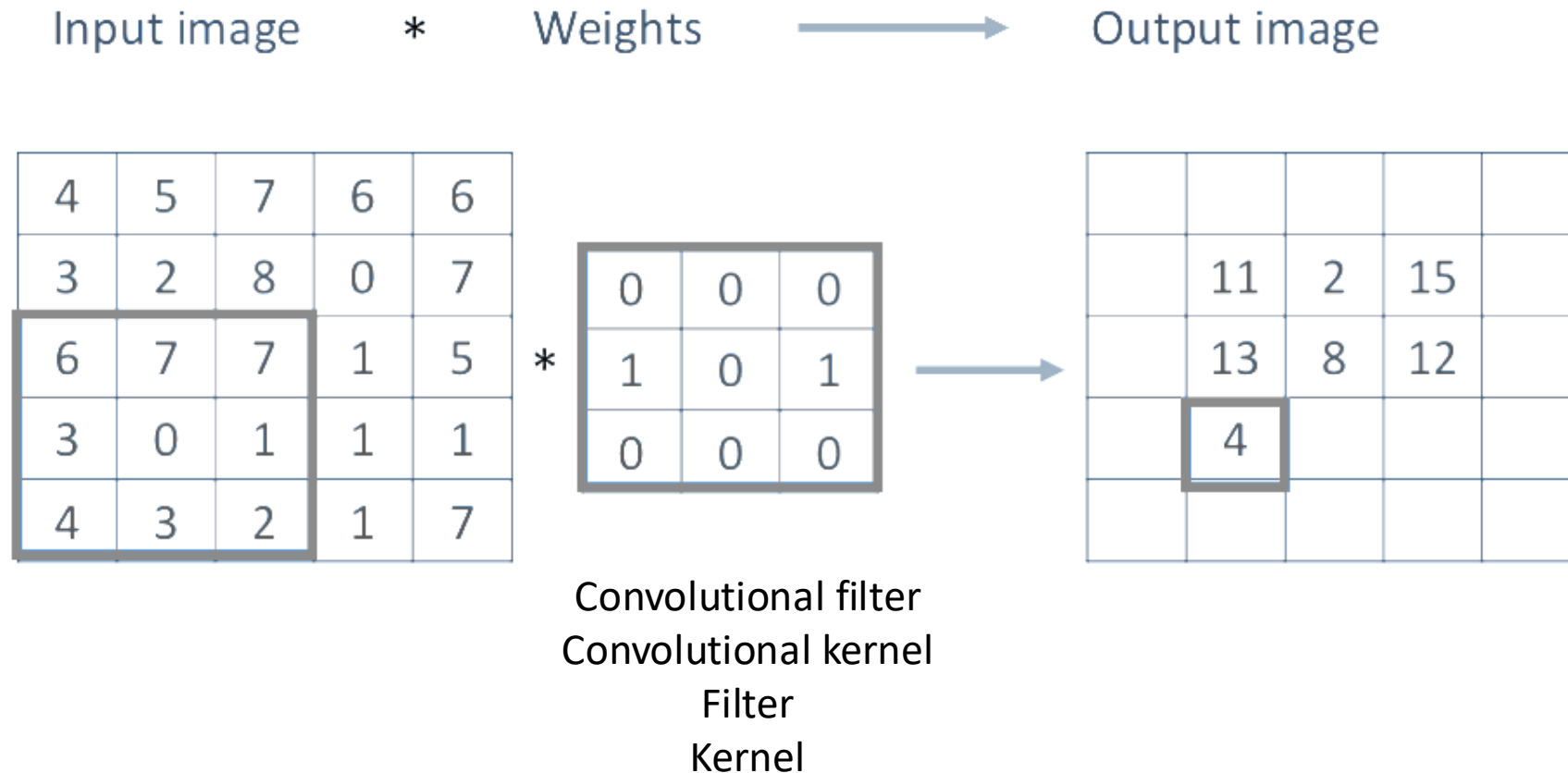
- The Convolution Operation



<http://www.cs.virginia.edu/~vicente/recognition/animation.gif>

Most important operation for Computer Vision (*)

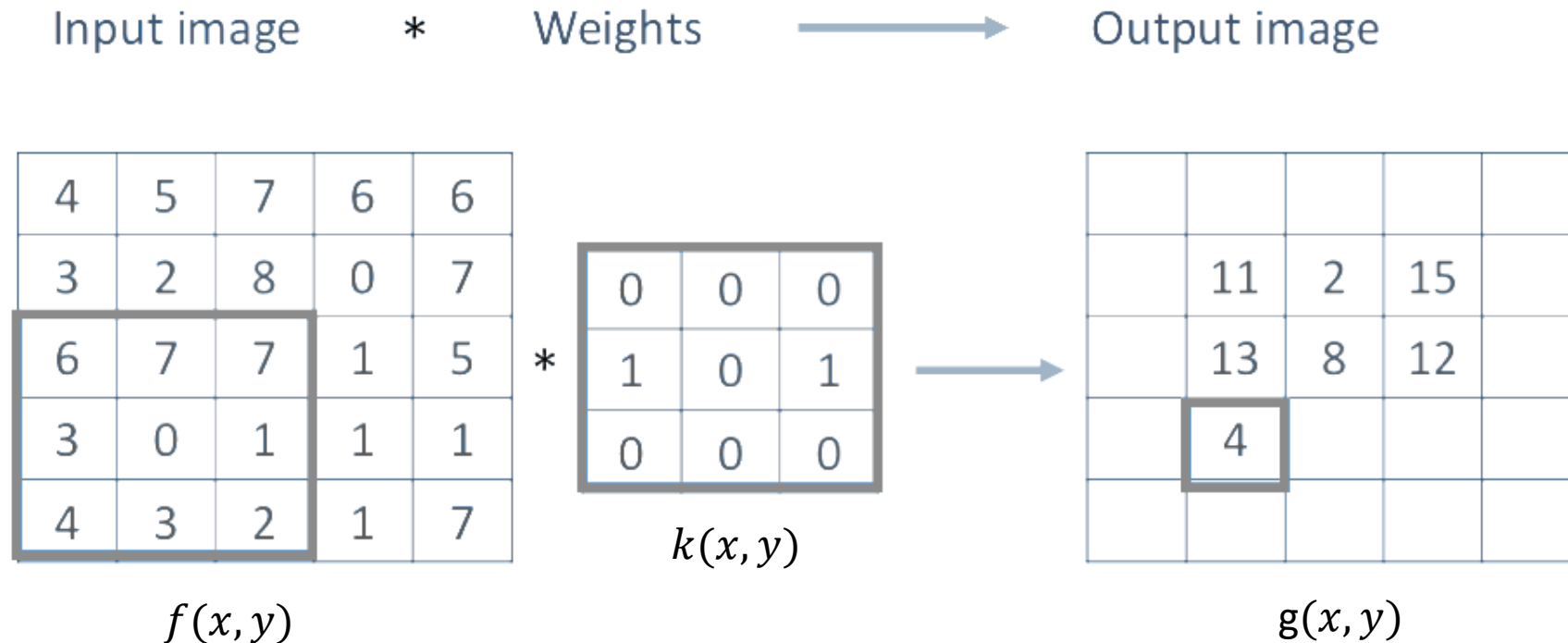
- The Convolution Operation



(*) Maybe

Most important operation for Computer Vision (*)

- The Convolution Operation

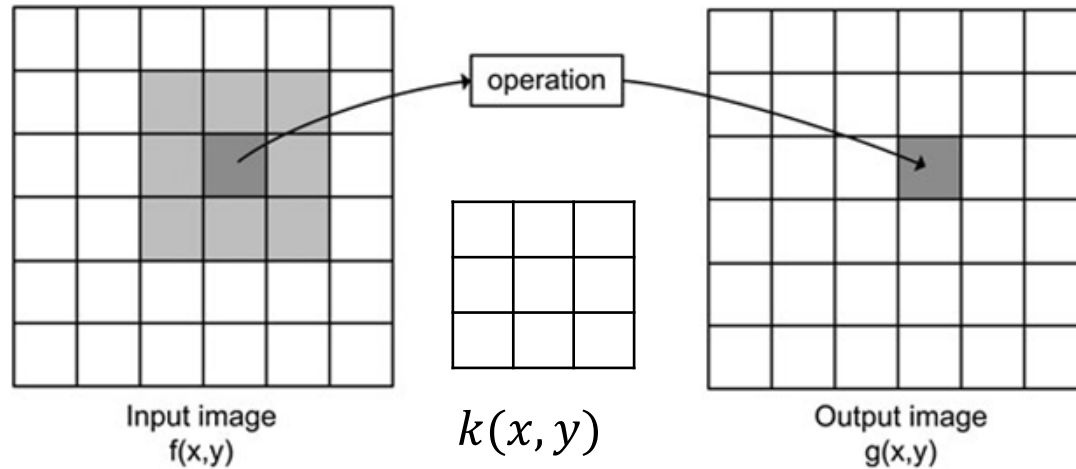


(*) Maybe

$$g(x, y) = \sum_v \sum_u k(u, v) f(x - u, y - v)$$

Image filtering: Convolution operator

e.g. mean filter

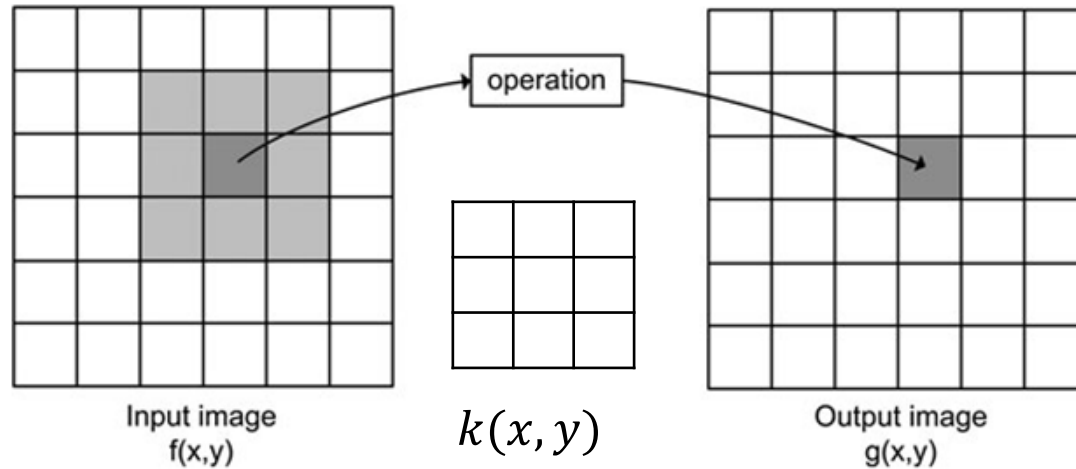


$$k(x, y) =$$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Image filtering: Convolution operator

e.g. mean filter



$$k(x,y) =$$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Example: box filter

$$\frac{1}{9} g[\cdot, \cdot]$$

1	1	1
1	1	1
1	1	1

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10							

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20						

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20	30					

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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	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	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

$h[\cdot, \cdot]$

	0	10	20	30	30				

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20	30	30				

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	0	10	20	30	30				
						?			
				50					

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$f[\cdot, \cdot]$

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

$h[\cdot, \cdot]$

	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	

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

$$\frac{1}{9} g[\cdot, \cdot]$$

1	1	1
1	1	1
1	1	1

Image filtering: e.g. Mean Filter

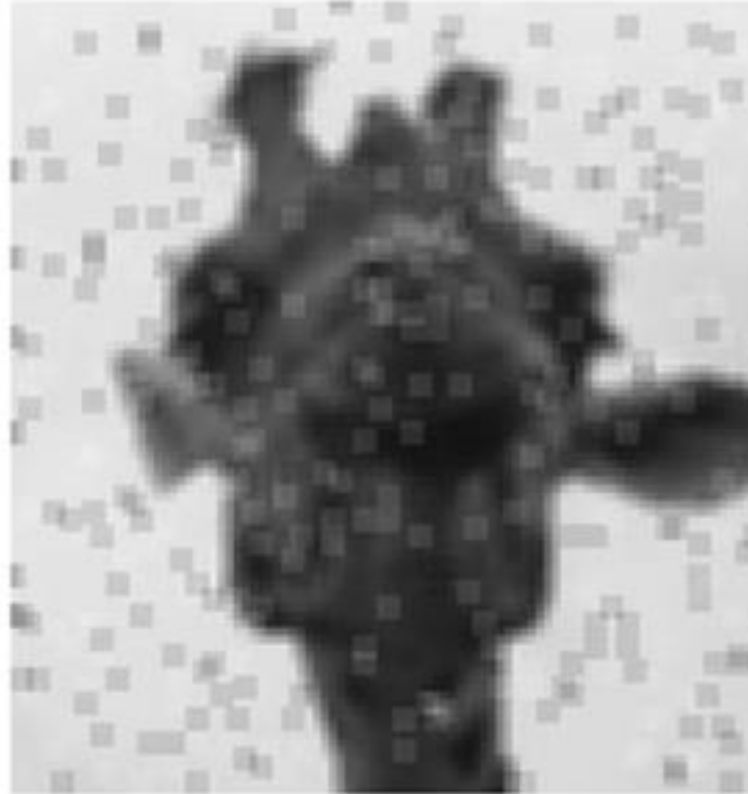
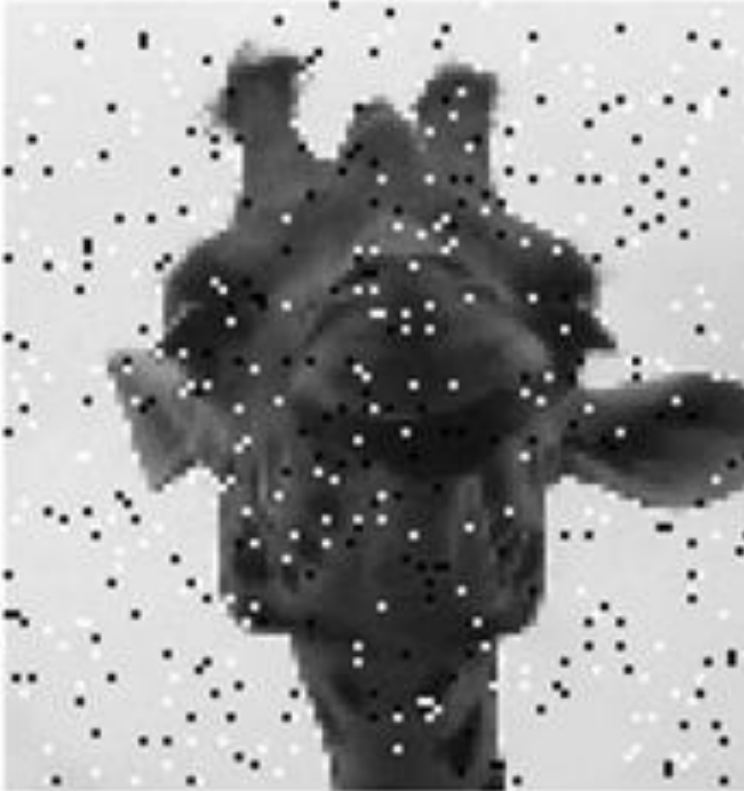
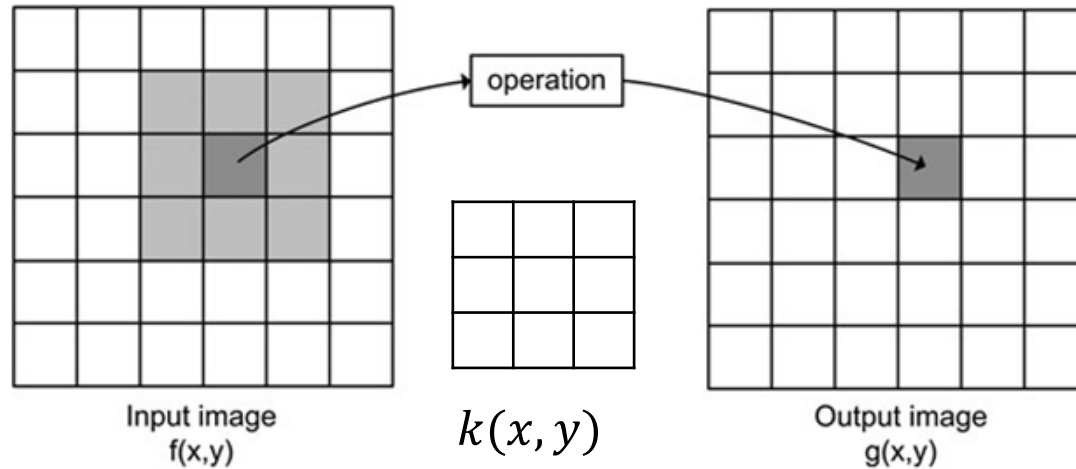
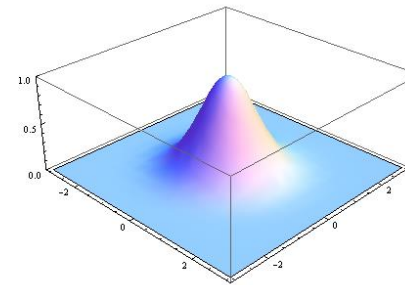


Image filtering: Convolution operator

Important filter: gaussian filter (gaussian blur)



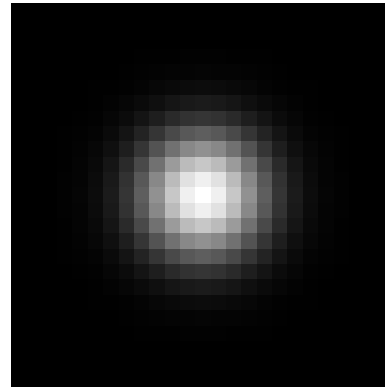
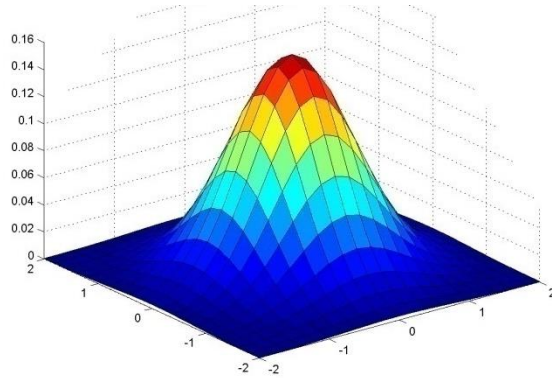
$$k(x,y) =$$



1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness



0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

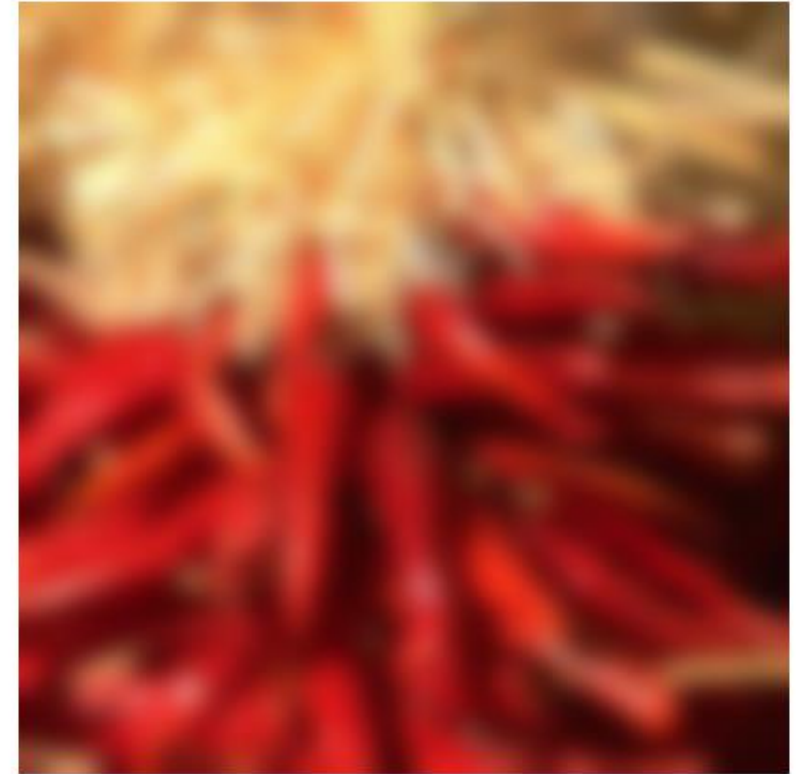
$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Image filtering: Convolution operator e.g. gaussian filter (gaussian blur)



Practical matters

- What about near the edge?
 - the filter window falls off the edge of the image
 - need to extrapolate
 - methods:
 - clip filter (black)
 - wrap around
 - copy edge
 - reflect across edge



Questions?