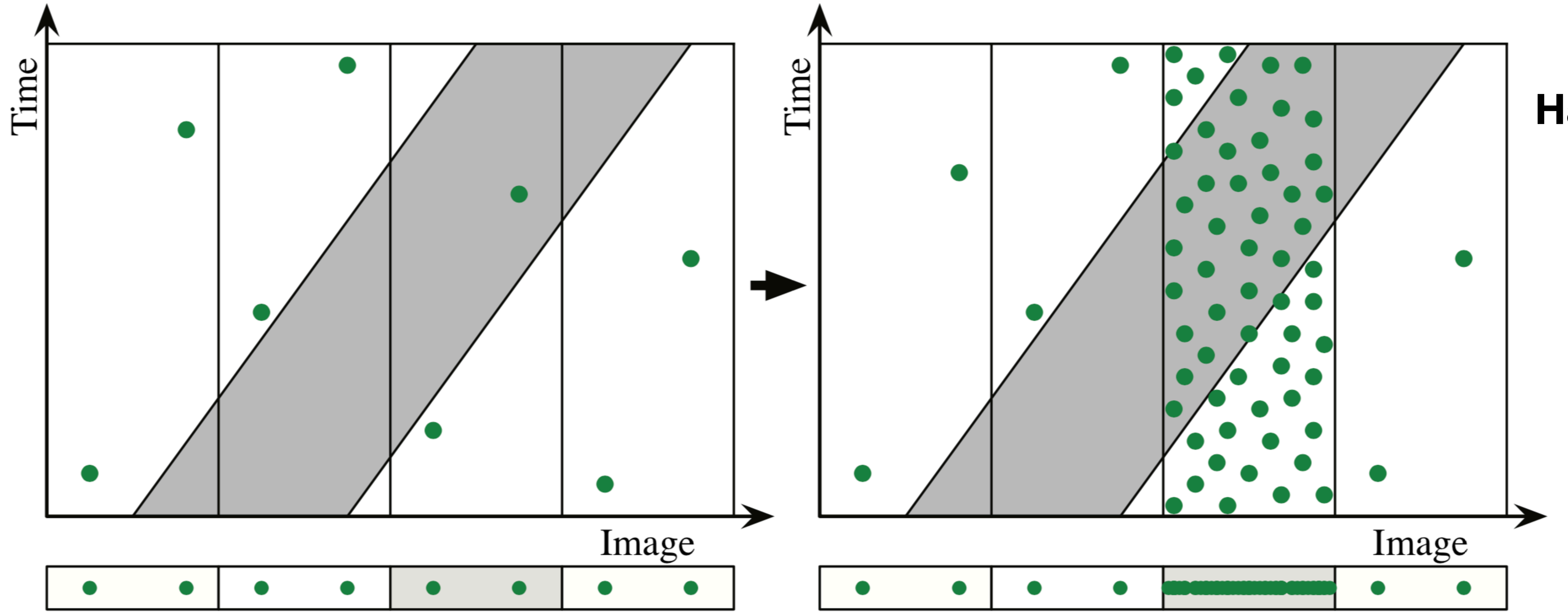
Denoising Algorithms: Path to Neural Networks I

Image courtesy Bako et al. [2017]

Philipp Slusallek *Karol Myszkowski* ***Gurprit Singh***

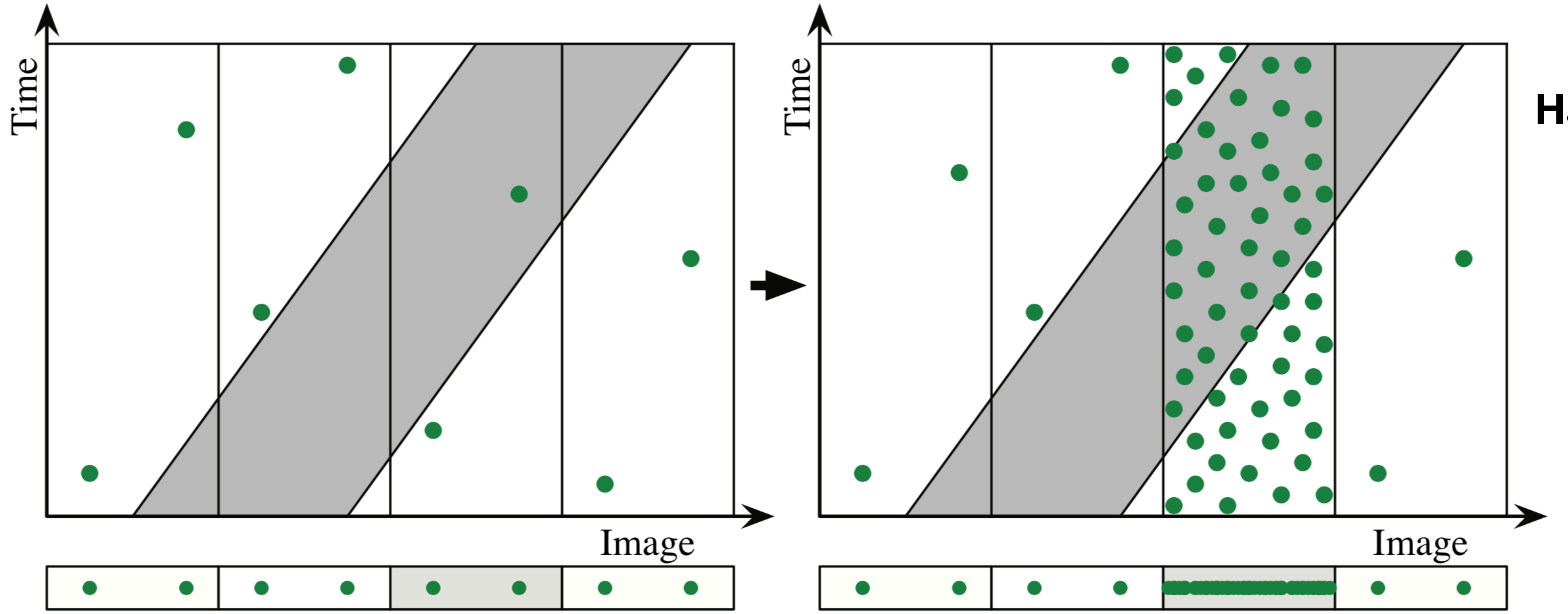
Previous Lecture Overview

Image-space Adaptive Sampling



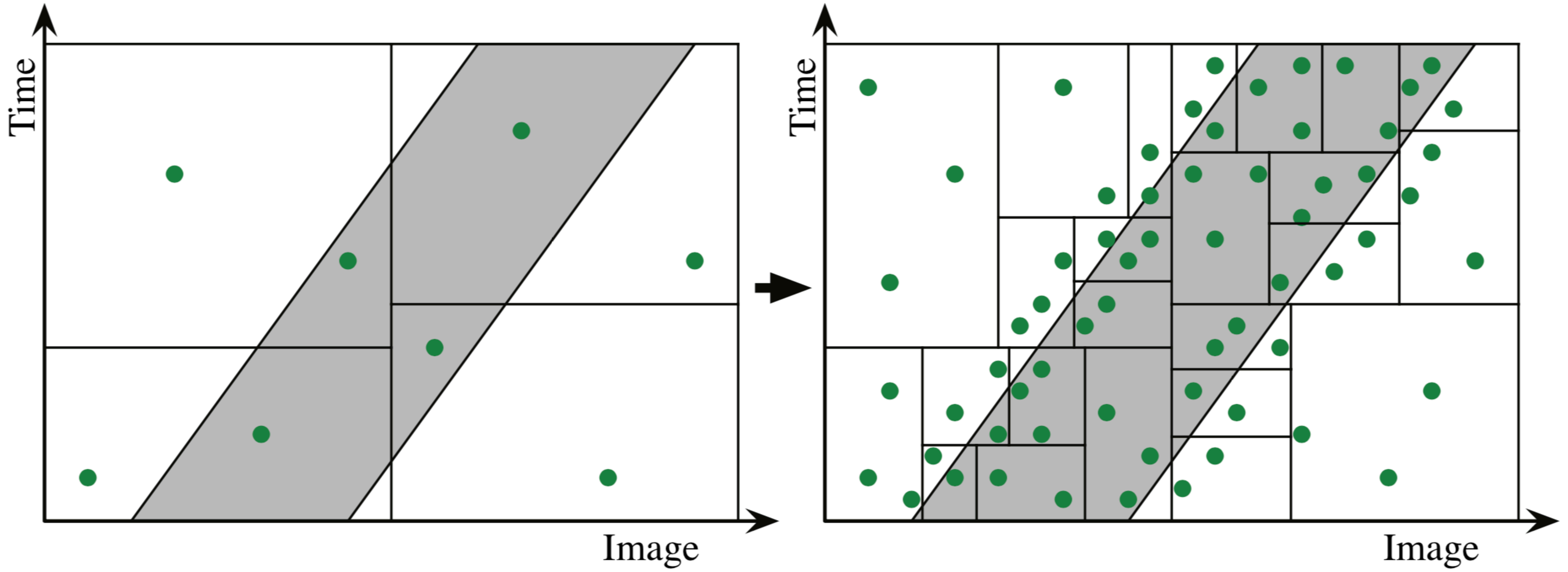
Hachisuka et al. [2008]

Image-space Adaptive Sampling

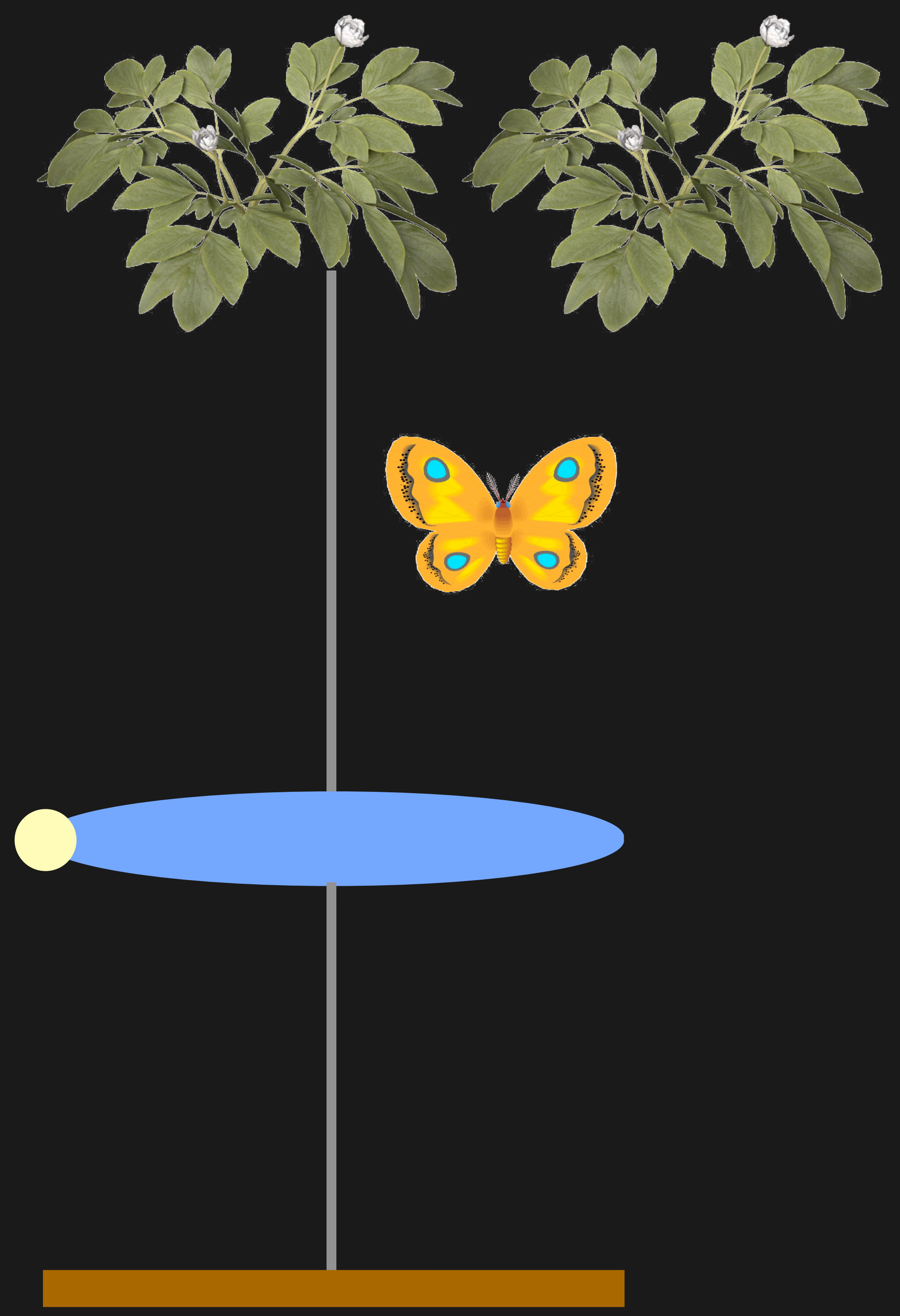


Hachisuka et al. [2008]

Multidimensional Adaptive Sampling



Depth of field



Slide from Jakko Lehtinen


Lens u
Screen x
Slide from Jakko Lehtinen

Visibility: SameSurface

The trajectories of samples originating from a single **apparent surface** never intersect.

Slide from Jakko Lehtinen

Introduction
Denoising using Data

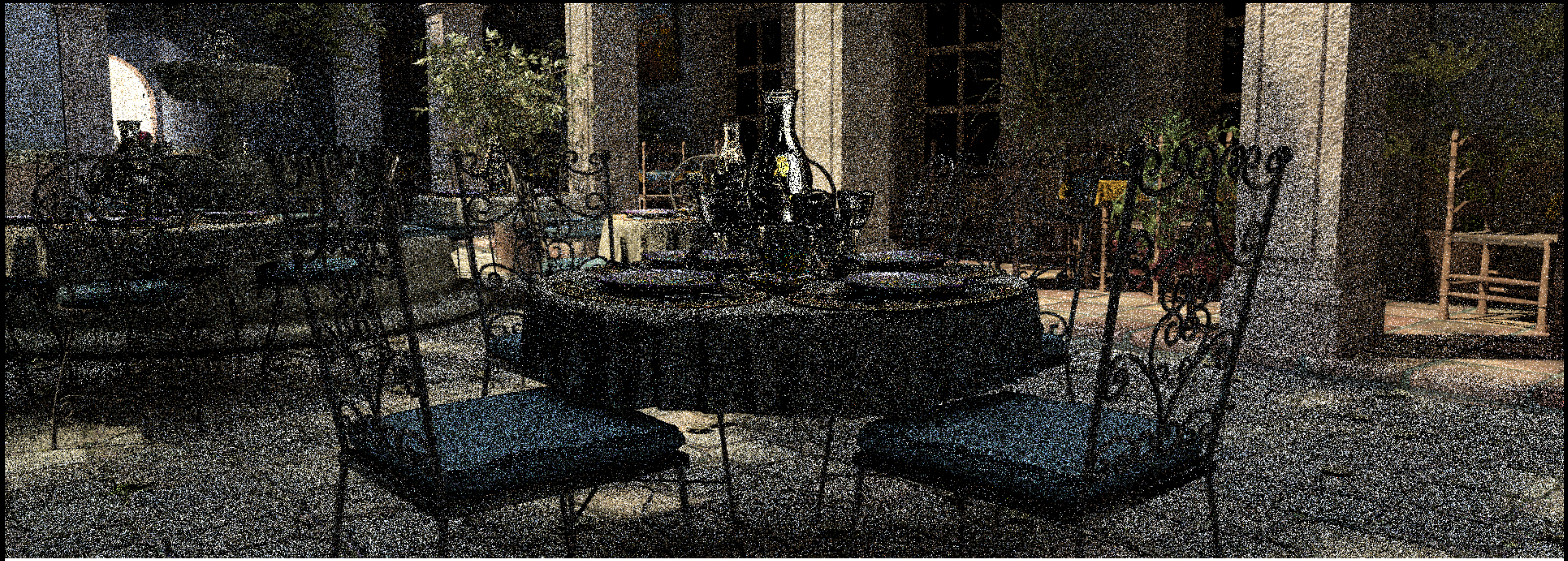
Path to Machine Learning

MLP based Denoising

CNN based Denosing
(Next lecture)

Filtering Monte Carlo Noise From Random Parameters

Sen and Darabi [2012]


input Monte Carlo (8 samples/pixel)


after RPF (8 samples/pixel)

High-dimensional Monte Carlo Integration

$$I(i,j) = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \cdots \int_{-1}^{1} \int_{-1}^{1} \int_{t_0}^{t_1} f(x, y, \cdots, u, v, t)\, dt\, dv\, du \cdots dy\, dx$$

(a) Input MC (8 spp) **(b)** Dependency on (u, v) **(c)** Our approach (RPF)

Parameters in Monte Carlo estimator

Random parameters: $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$

Color: $\mathbf{c}_i \Leftarrow f(\underbrace{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}}_{\text{screen position}}; \underbrace{\mathbf{r}_{i,1}, \mathbf{r}_{i,2}, \ldots, \mathbf{r}_{i,n}}_{\text{random parameters}})$

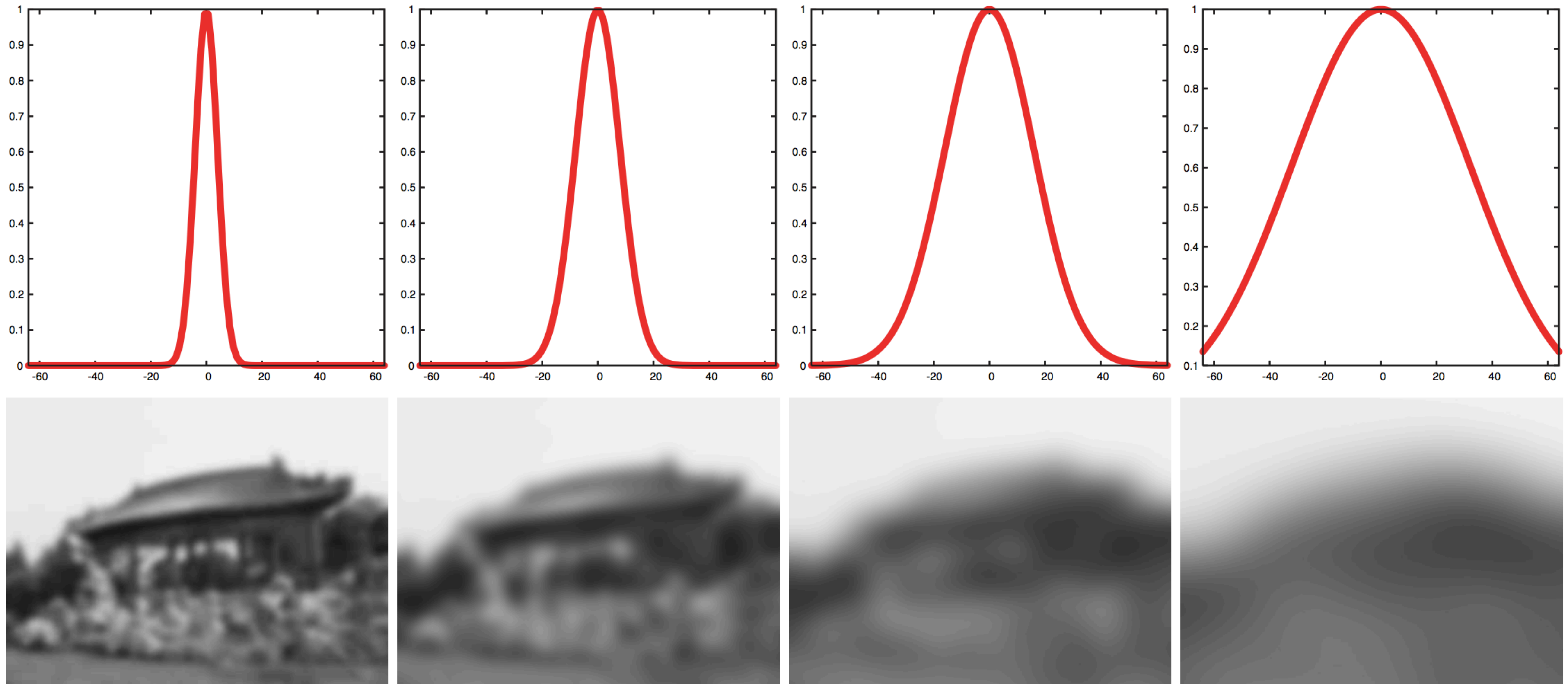
Random Parameters Classification

Random parameter for each pixel :

$$\mathbf{x}_i \Leftarrow f(\mathbf{p}_{i,1}, \mathbf{p}_{i,2}; \mathbf{r}_{i,1}, \mathbf{r}_{i,2}, \ldots, \mathbf{r}_{i,n})$$

$$\mathbf{x}_i = \{\underbrace{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}}_{\text{screen position}}; \underbrace{\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,n}}_{\text{random parameters}}; \underbrace{\mathbf{f}_{i,1}, \ldots, \mathbf{f}_{i,m}}_{\text{scene features}}; \underbrace{\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \mathbf{c}_{i,3}}_{\text{sample color}}\}$$

Gaussian Filtering



$\sigma = 4$ $\sigma = 8$ $\sigma = 16$ $\sigma = 32$

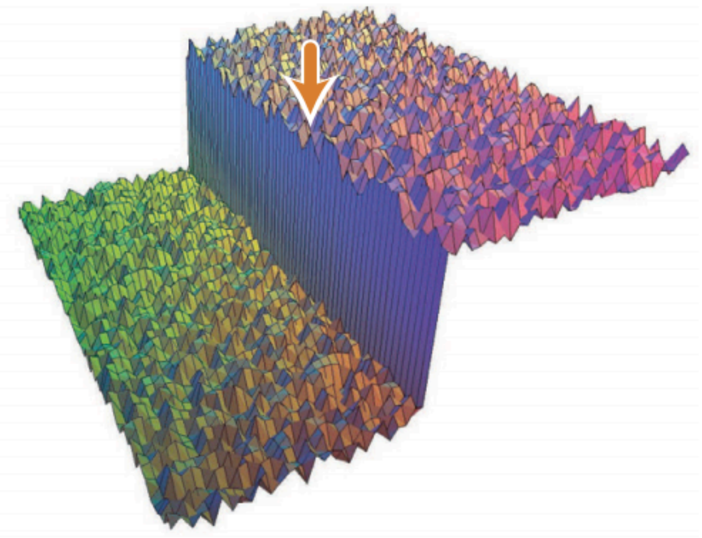
$$GC[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_\sigma(\|\mathbf{p} - \mathbf{q}\|)\, I_{\mathbf{q}}, \qquad G_\sigma(x) = \frac{1}{2\pi\,\sigma^2} \exp\left(-\frac{x^2}{2\,\sigma^2}\right)$$

Bilateral Filtering

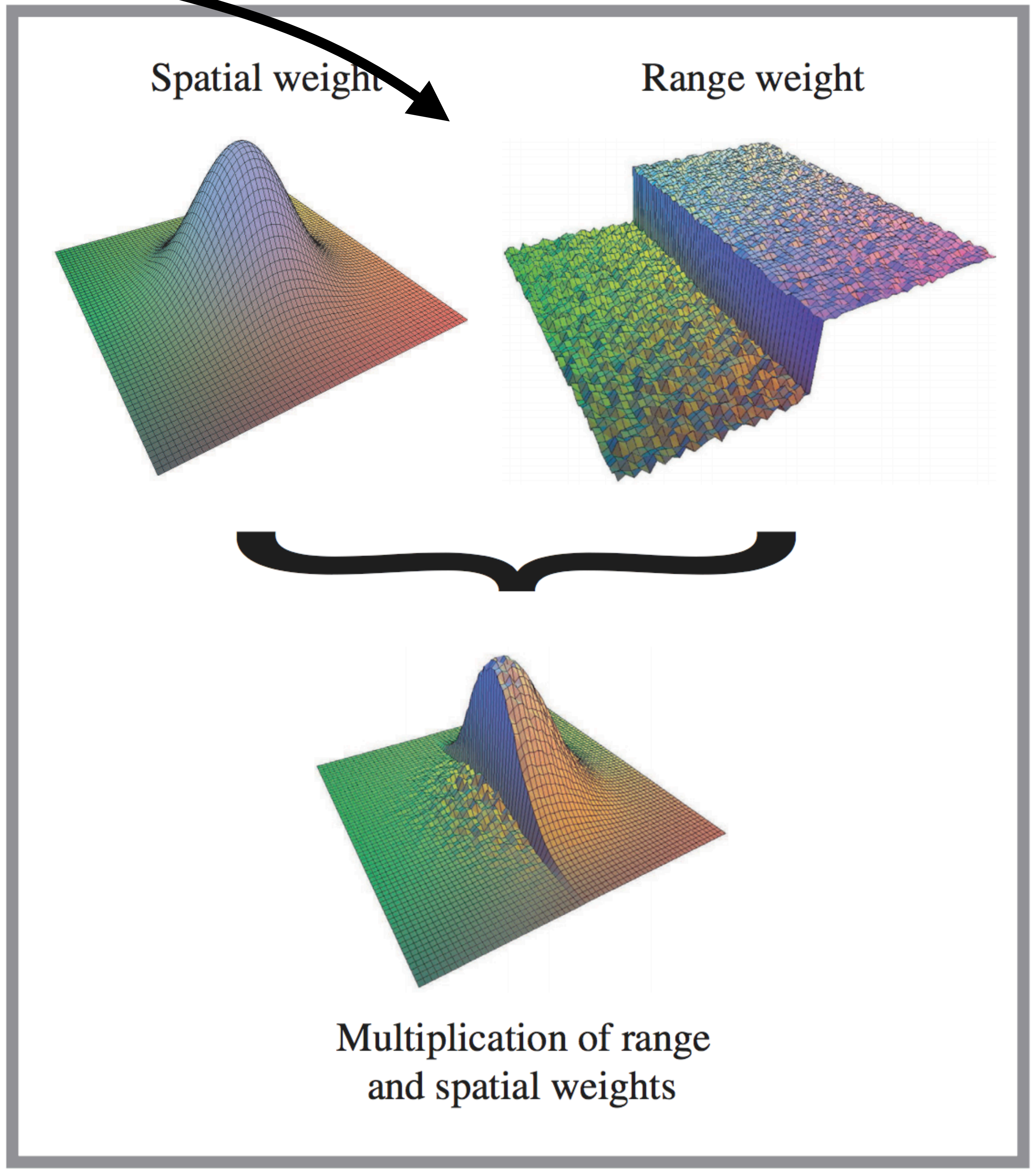
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) \, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) \, I_{\mathbf{q}}$$

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) \, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)$$

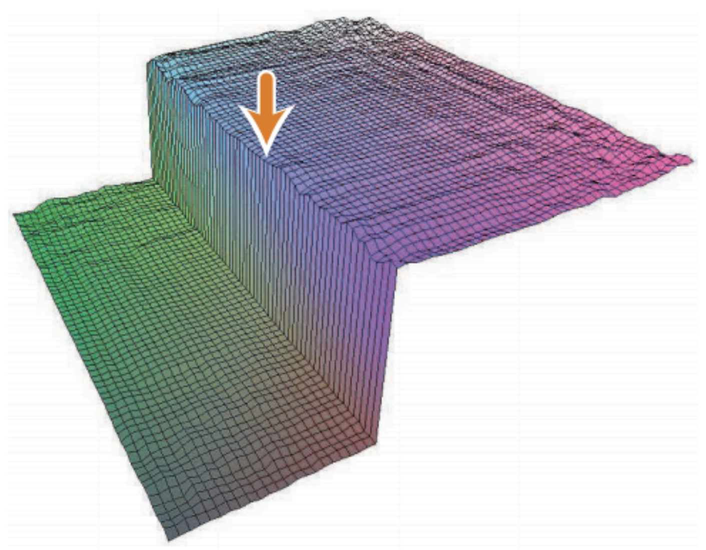
Input



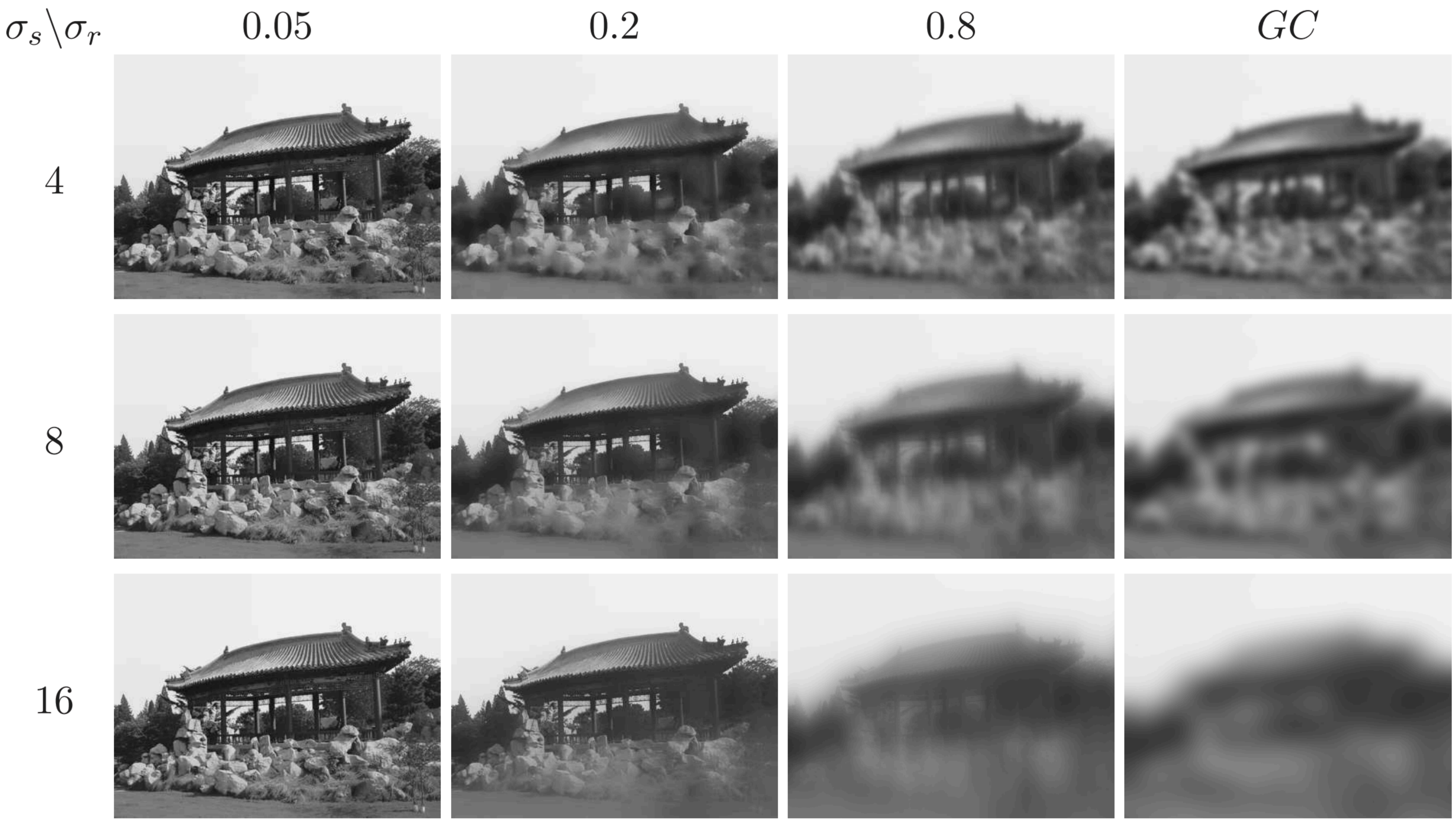
Bilateral filter weights at the central pixel



Result



Bilateral vs Gaussian Filtering



Bilateral Filtering of Features

$$w_{ij} = \exp[-\frac{1}{2\sigma_{\mathbf{p}}^2} \sum_{1 \le k \le 2} (\bar{\mathbf{p}}_{i,k} - \bar{\mathbf{p}}_{j,k})^2] \times$$

$$\exp[-\frac{1}{2\sigma_{\mathbf{c}}^2} \sum_{1 \le k \le 3} \alpha_k (\bar{\mathbf{c}}_{i,k} - \bar{\mathbf{c}}_{j,k})^2] \times$$

$$\exp[-\frac{1}{2\sigma_{\mathbf{f}}^2} \sum_{1 \le k \le m} \beta_k (\bar{\mathbf{f}}_{i,k} - \bar{\mathbf{f}}_{j,k})^2],$$

Dependency on Random Parameters

Input Monte Carlo (8 spp) | Dependency of color on random parameters ($D_{\mathbf{c}}^{\mathbf{r}}$) | Dependency of color on screen position ($D_{\mathbf{c}}^{\mathbf{p}}$) | Fractional dependency on random parameters ($W_{\mathbf{c}}^{\mathbf{r}}$) | Our approach (RPF) | Reference MC (512 spp)

Bilateral Weights

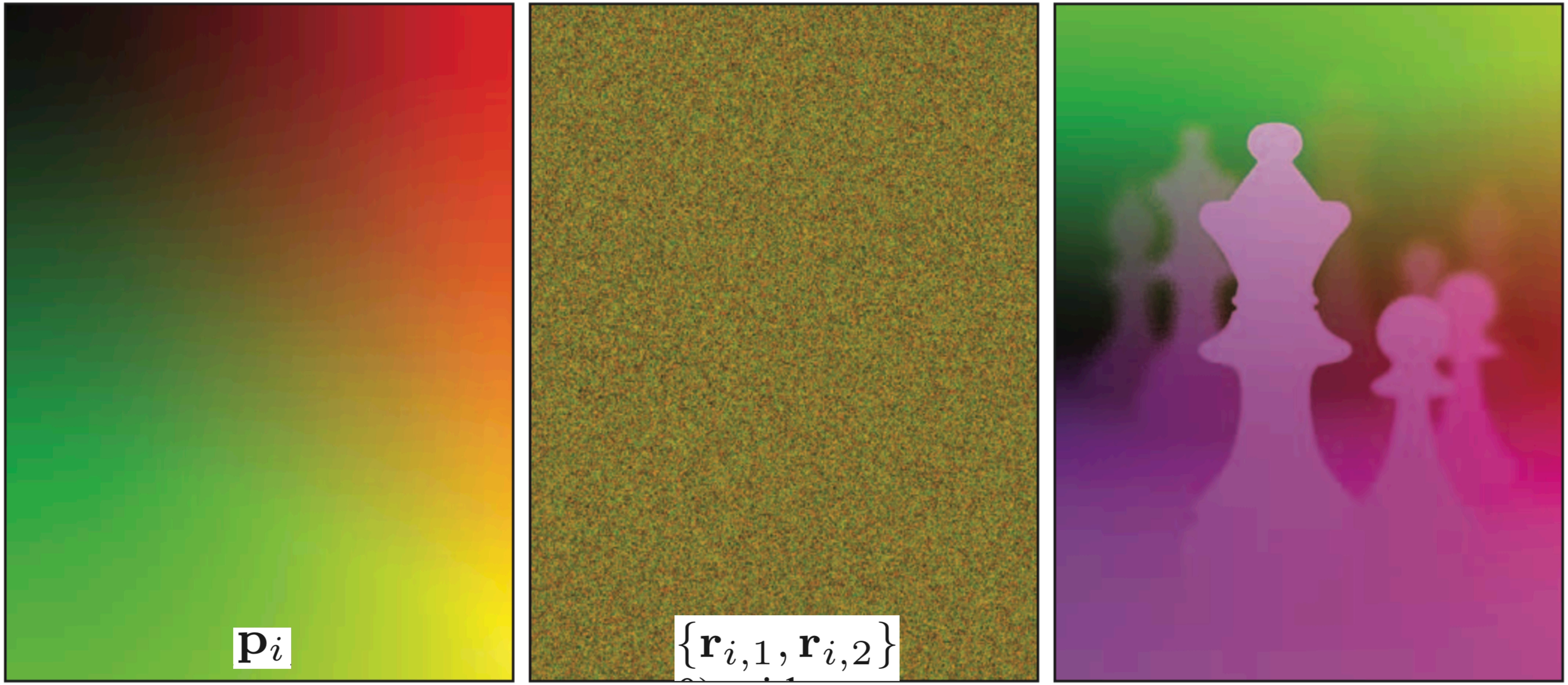
$$W^{\mathbf{r}}_{\mathbf{f},k} = \frac{D^{\mathbf{r}}_{\mathbf{f},k}}{D^{\mathbf{r}}_{\mathbf{f},k} + D^{\mathbf{p}}_{\mathbf{f},k}}$$

$$\beta_k = 1 - W^{\mathbf{r}}_{\mathbf{f},k}$$

$$W^{\mathbf{r}}_{\mathbf{c},k} = \frac{D^{\mathbf{r}}_{\mathbf{c},k}}{D^{\mathbf{r}}_{\mathbf{c},k} + D^{\mathbf{p}}_{\mathbf{c},k}}$$

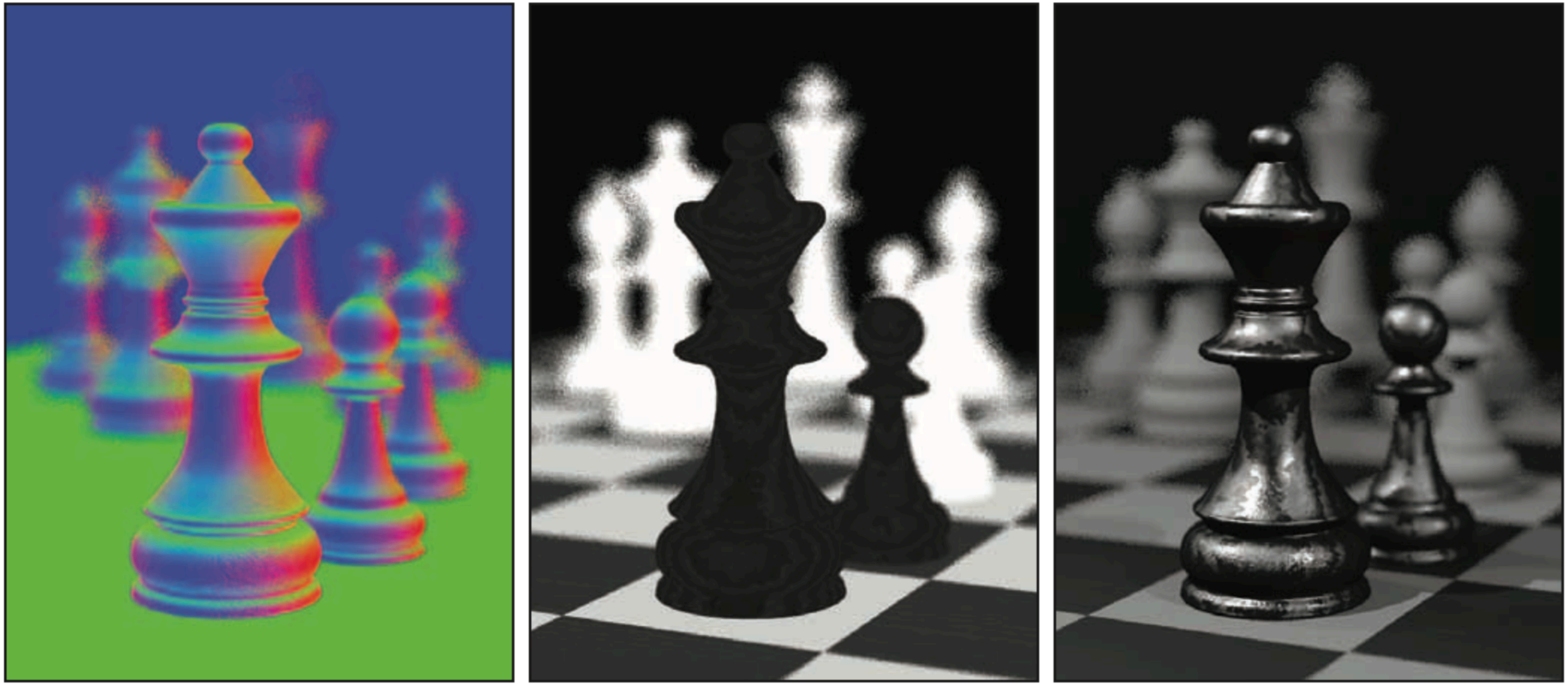
$$\alpha_k = 1 - W^{\mathbf{r}}_{\mathbf{c},k}.$$

Pixels,Random Params,Features



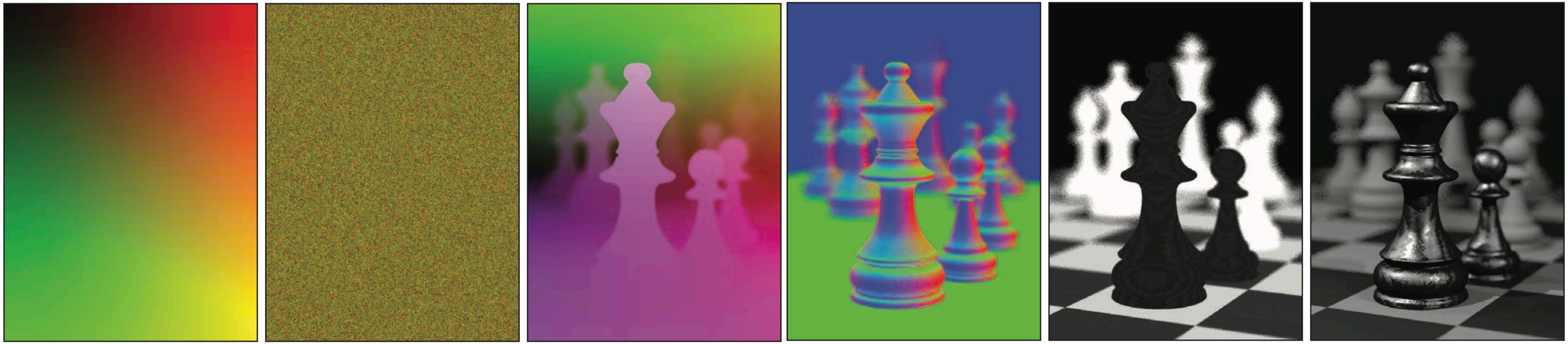
(a) Screen position (b) Random parameters (c) World space coords.

Pixels,Random Params,Features



(d) Surface normals (e) Texture value (f) Sample color

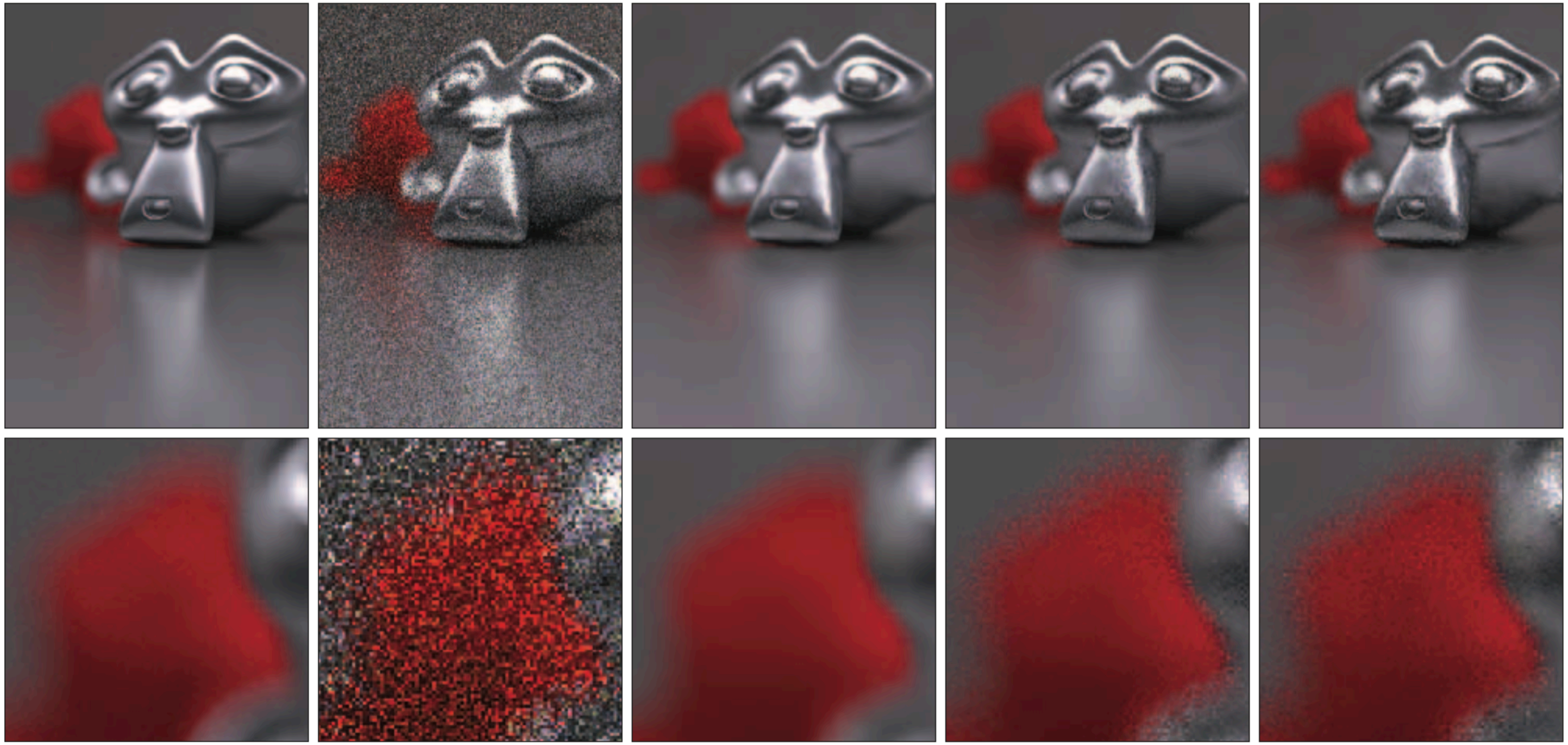
Pixels,Random Params,Features



(a) Screen position (b) Random parameters (c) World space coords. (d) Surface normals (e) Texture value (f) Sample color

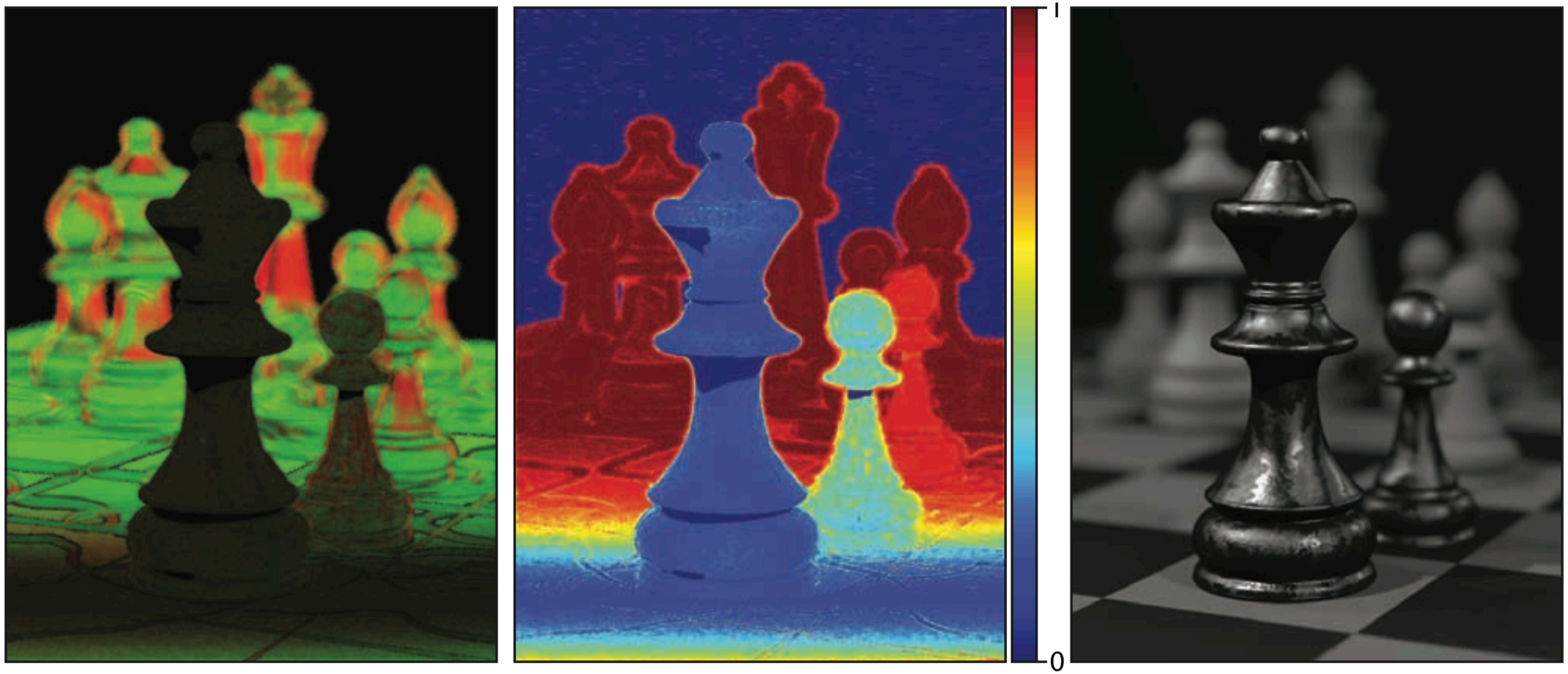
The algorithm computes the statistical dependency of **(c-f)** on the random parameters in (**b**)

Random Parameter Filtering



(a) Reference **(b)** MC Input **(c)** RPF **(d)** no clustering **(e)** no DoF params

Random Parameter Filtering



(a) $W^{\mathbf{r},1}_{\mathbf{c},k}$ and $W^{\mathbf{r},2}_{\mathbf{c},k}$ **(b)** $W^{\mathbf{r}}_{\mathbf{c},k}$ **(c)** Our output (RPF)

Statistical Dependency

Mutual information between two random variables:

$$\mu(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

where, these probabilities are computed over
the neighborhood of samples around a given pixel

Statistical Dependency

Functional dependency of the k-th scene parameter:

$$D^{\mathbf{r}}_{\mathbf{f},k} = \sum_{1 \le l \le n} D^{\mathbf{r},l}_{\mathbf{f},k} = \sum_{1 \le l \le n} \mu(\bar{\mathbf{f}}_{\mathcal{N},k}; \bar{\mathbf{r}}_{\mathcal{N},l})$$

$$D^{\mathbf{p}}_{\mathbf{f},k} = \sum_{1 \le l \le 2} D^{\mathbf{p},l}_{\mathbf{f},k} = \sum_{1 \le l \le 2} \mu(\bar{\mathbf{f}}_{\mathcal{N},k}; \bar{\mathbf{p}}_{\mathcal{N},l}),$$

$$D^{\mathbf{r}}_{\mathbf{c},k} = \sum_{1 \le l \le n} D^{\mathbf{r},l}_{\mathbf{c},k} = \sum_{1 \le l \le n} \mu(\bar{\mathbf{c}}_{\mathcal{N},k}; \bar{\mathbf{r}}_{\mathcal{N},l}),$$

$$D^{\mathbf{p}}_{\mathbf{c},k} = \sum_{1 \le l \le 2} D^{\mathbf{p},l}_{\mathbf{c},k} = \sum_{1 \le l \le 2} \mu(\bar{\mathbf{c}}_{\mathcal{N},k}; \bar{\mathbf{p}}_{\mathcal{N},l}).$$

Statistical Dependency

$$D^{\mathbf{r}}_{\mathbf{f},k} = \sum_{1 \le l \le n} D^{\mathbf{r},l}_{\mathbf{f},k} = \sum_{1 \le l \le n} \mu(\bar{\mathbf{f}}_{\mathcal{N},k}; \bar{\mathbf{r}}_{\mathcal{N},l})$$

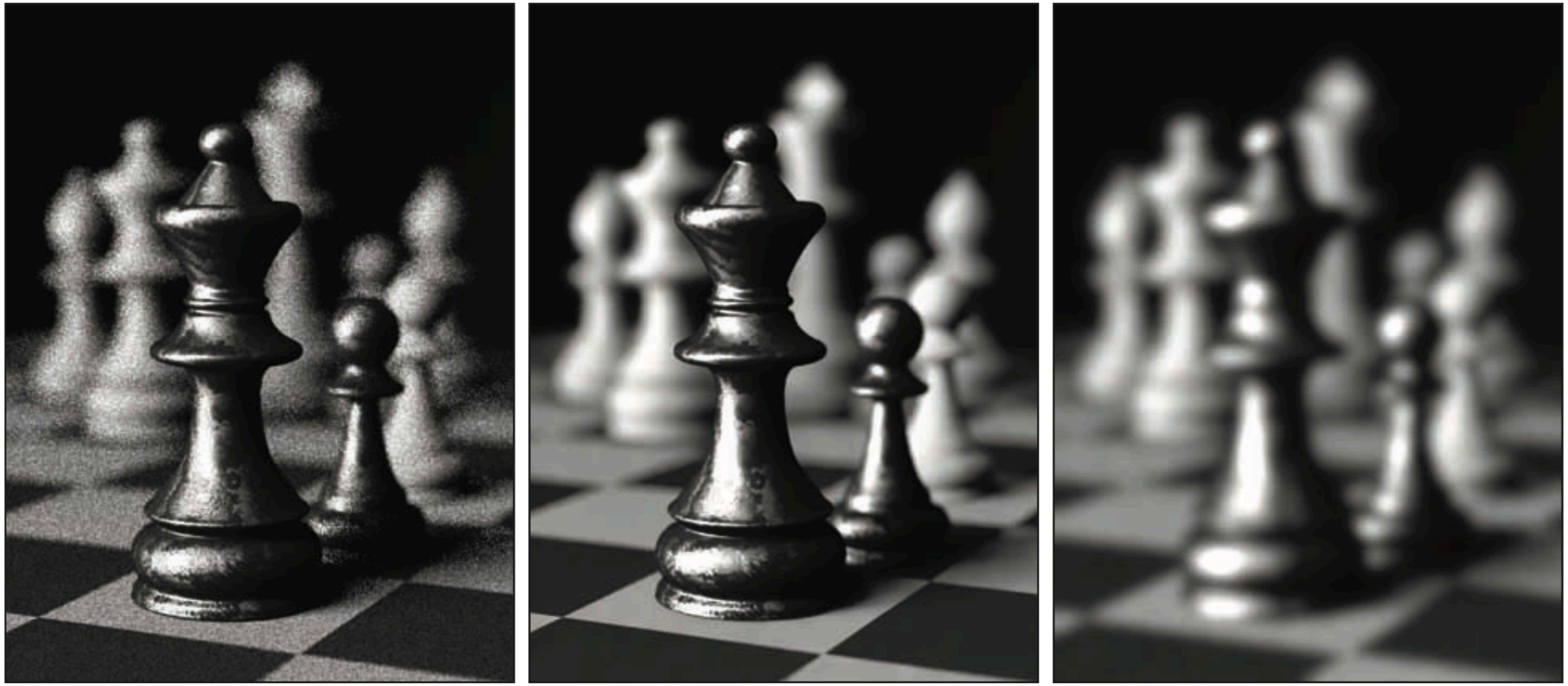
$$W^{\mathbf{f},k}_{\mathbf{c}} = \frac{D^{\mathbf{f},k}_{\mathbf{c}}}{D^{\mathbf{r}}_{\mathbf{c}} + D^{\mathbf{p}}_{\mathbf{c}} + D^{\mathbf{f}}_{\mathbf{c}}}$$

$$D^{\mathbf{r}}_{\mathbf{c}} = \sum_{1 \le k \le 3} D^{\mathbf{r}}_{\mathbf{c},k}, \quad D^{\mathbf{p}}_{\mathbf{c}} = \sum_{1 \le k \le 3} D^{\mathbf{p}}_{\mathbf{c},k}, \quad D^{\mathbf{f}}_{\mathbf{c}} = \sum_{1 \le k \le 3} D^{\mathbf{f}}_{\mathbf{c},k}$$

Weighted Average Bilateral Filtering

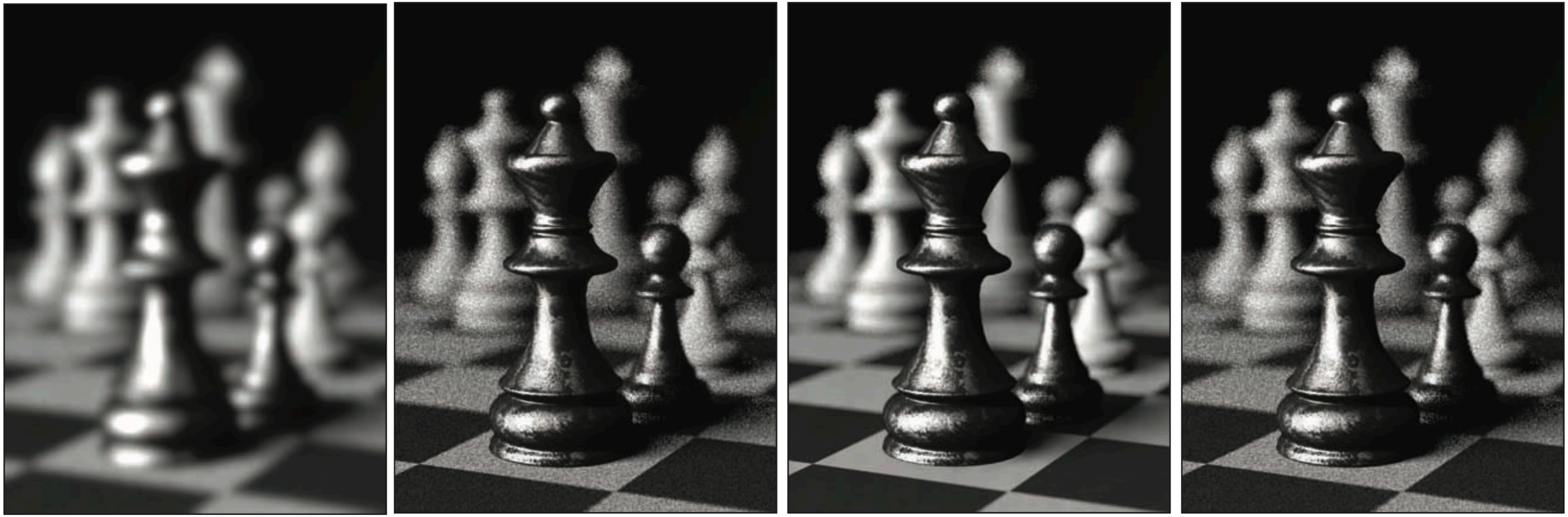
$$\mathbf{c}'_{i,k} = \frac{\sum_{j \in \mathcal{N}} w_{ij} \mathbf{c}_{j,k}}{\sum_{j \in \mathcal{N}} w_{ij}}$$

Results



(a) MC Input (8 spp) **(b)** Our approach (RPF) **(c)** $\alpha_k = 0, \beta_k = 0$

Results



) (c) $\alpha_k = 0, \beta_k = 0$ (d) $\alpha_k = 1, \beta_k = 0$ (e) $\alpha_k = 0, \beta_k = 1$ (f) $\alpha_k = 1, \beta_k = 1$

Results

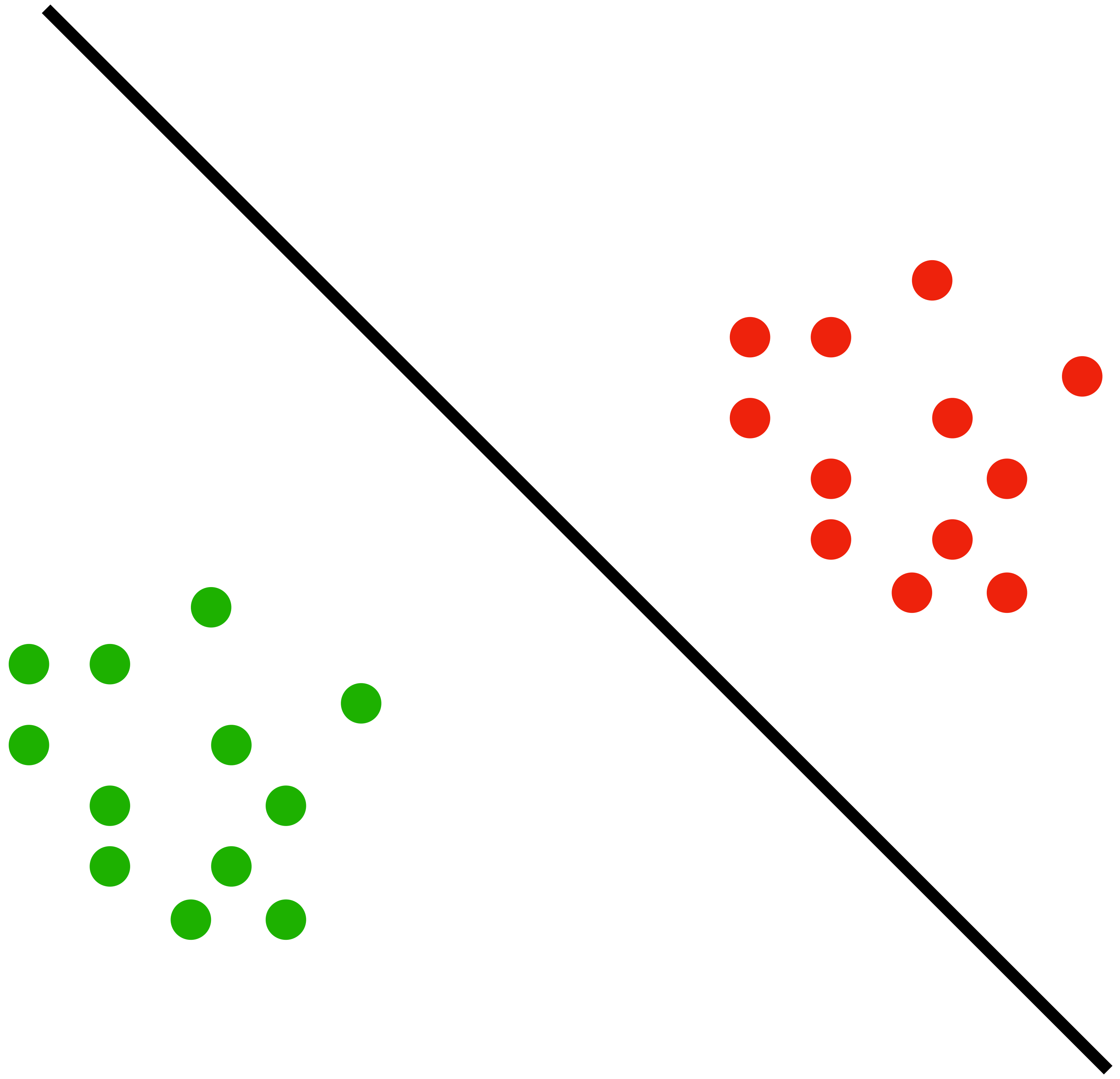
Reference (8,192 spp) | Input Monte Carlo (8 spp) | MDAS | AWR | À-Trous | Our approach (RPF)

Multi-Layer Perceptrons

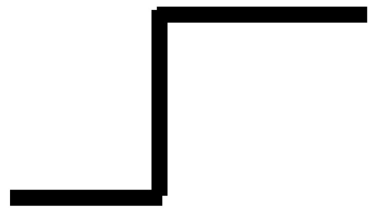
History of Neural Networks

- In 1943, McCulloch and Pitts created a computational model for neural networks
- In 1975, Werbos's back propagation algorithm generally accelerated the training of multi-layer networks.
- In 1980s, Recurrent Neural Networks were developed

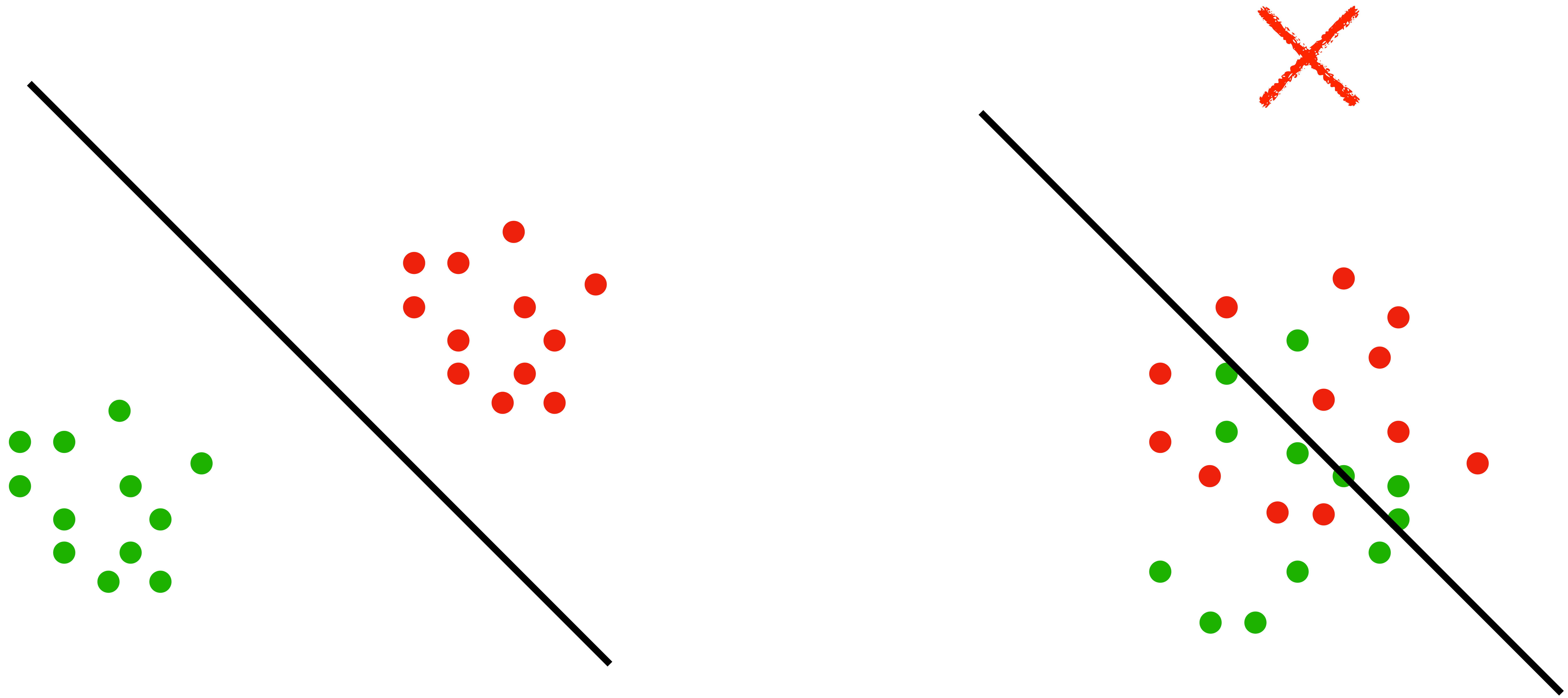
Classifiers



$$y_j = f(w_j x_j + b_j)$$

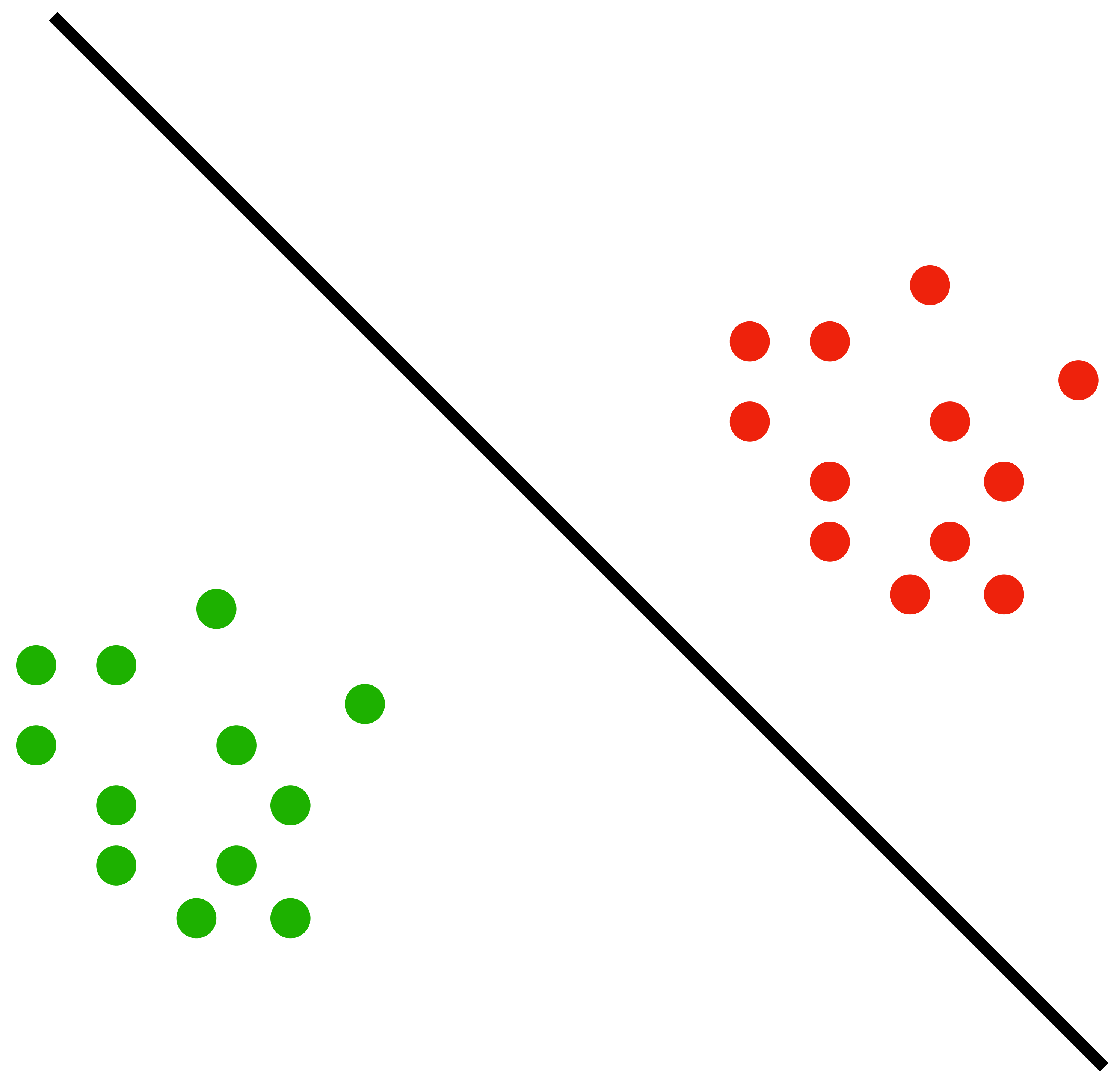


Classifiers

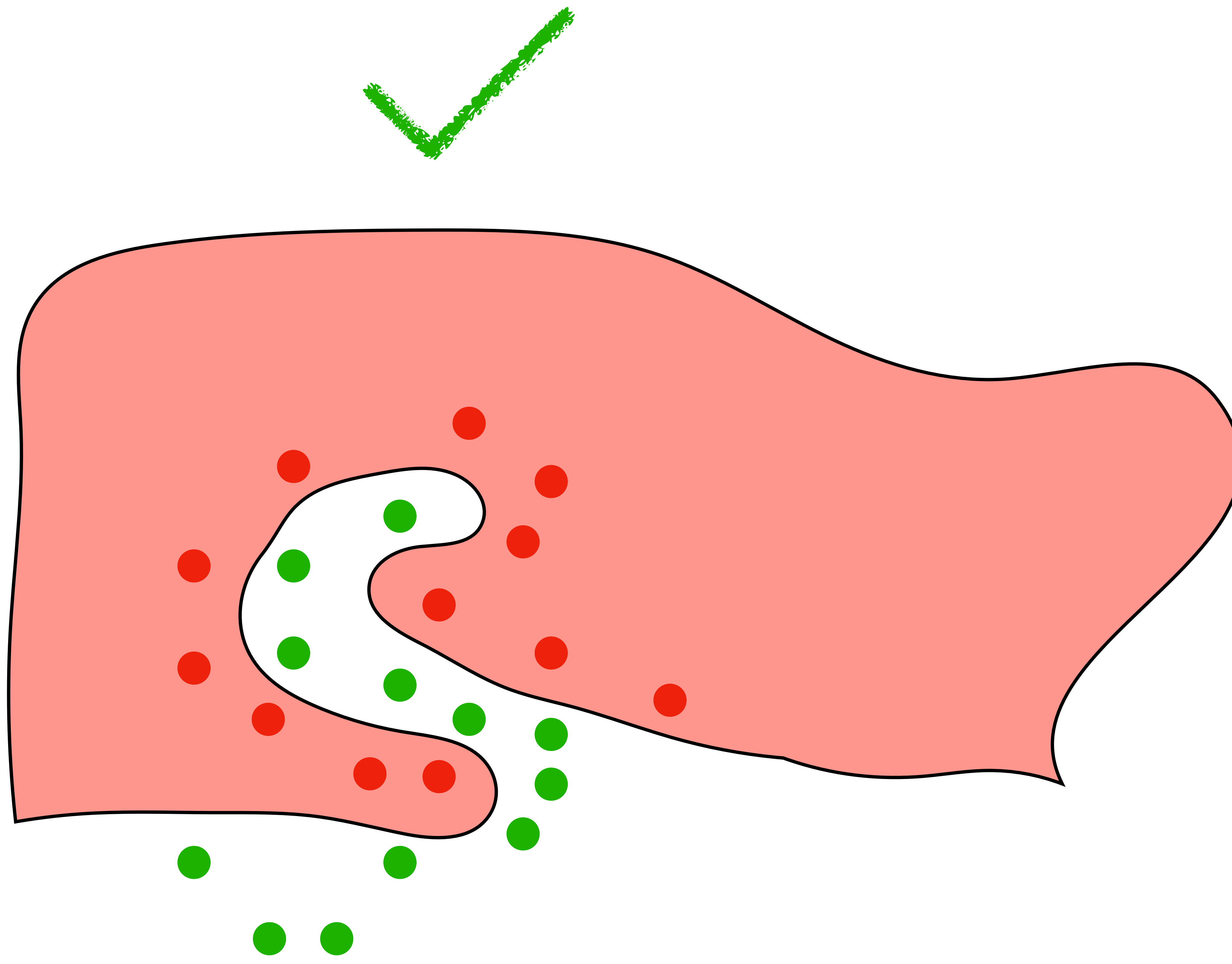


$$y_j = f(w_j x_j + b_j)$$

Complex Classifiers



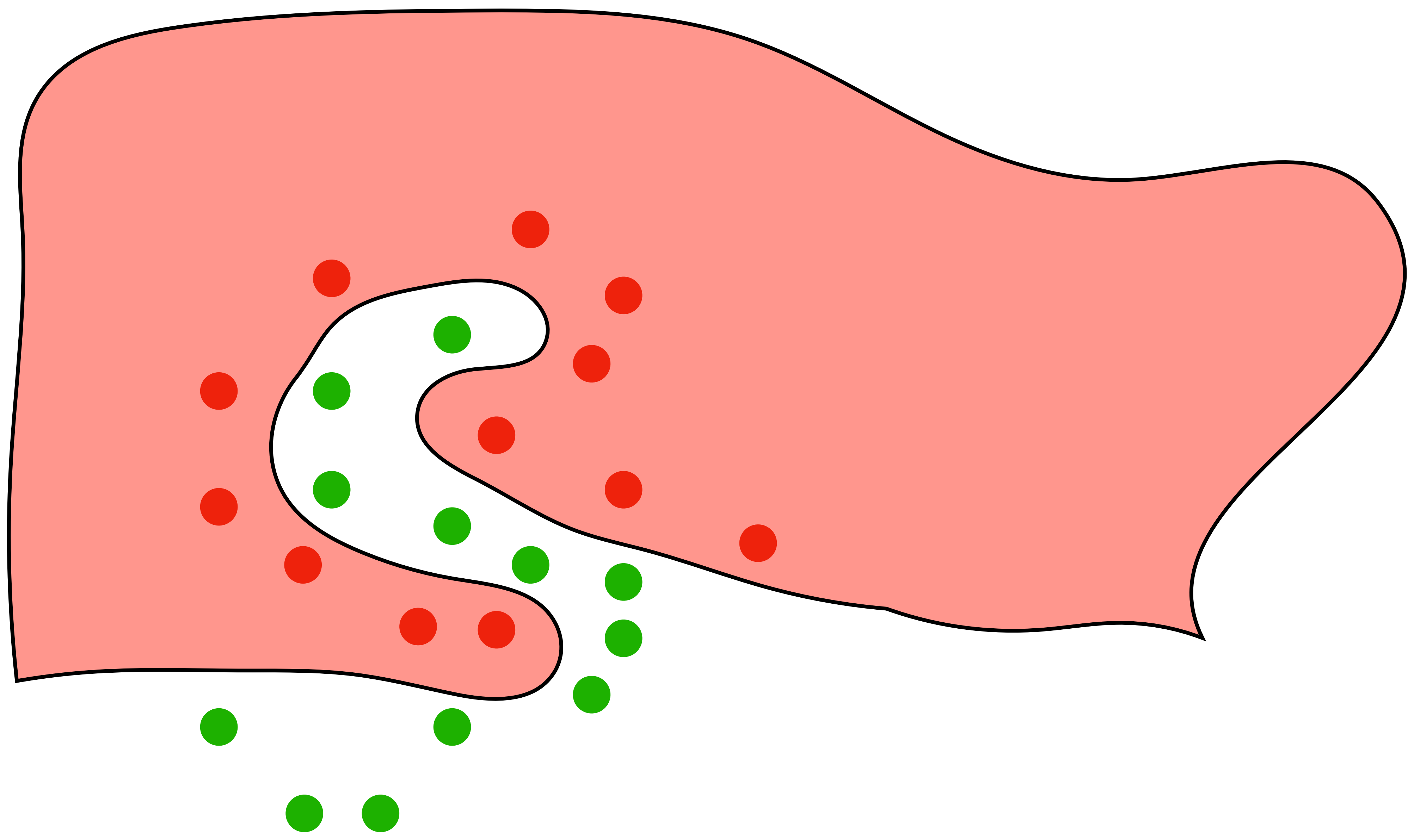
$$y_j = f(w_j x_j + b_j)$$



Complex classifier

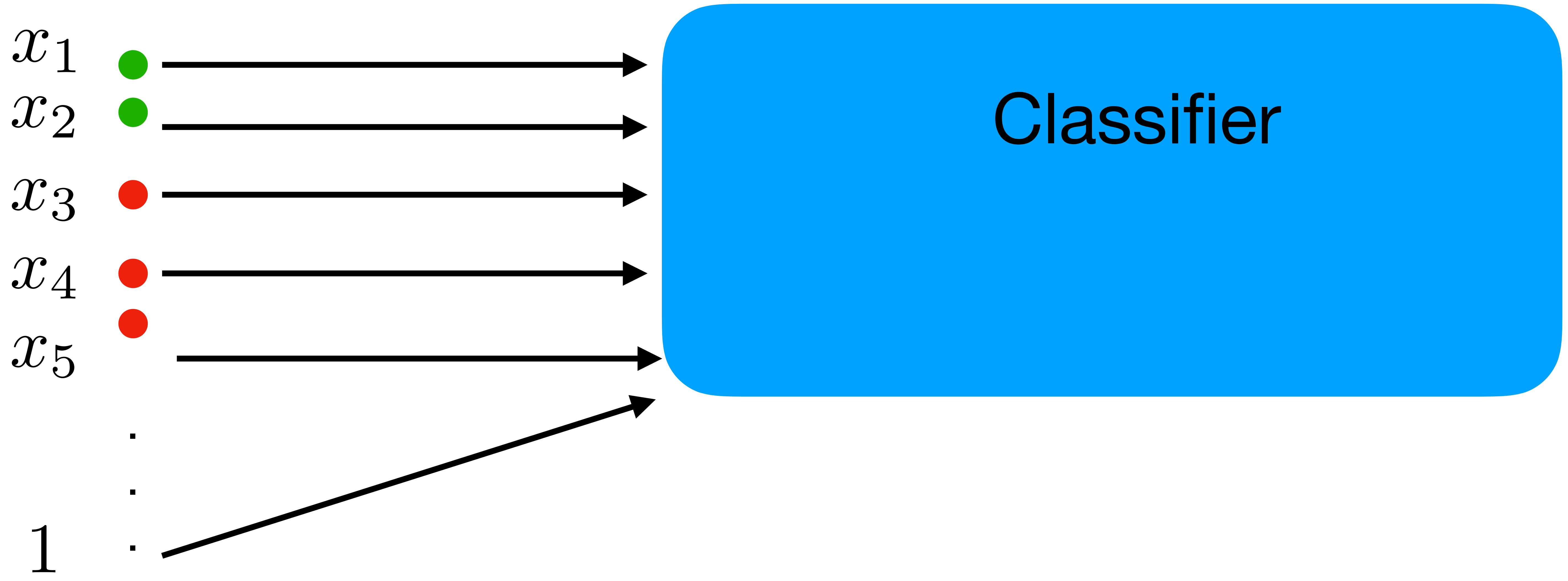
Complex Classifiers

Complex classifier

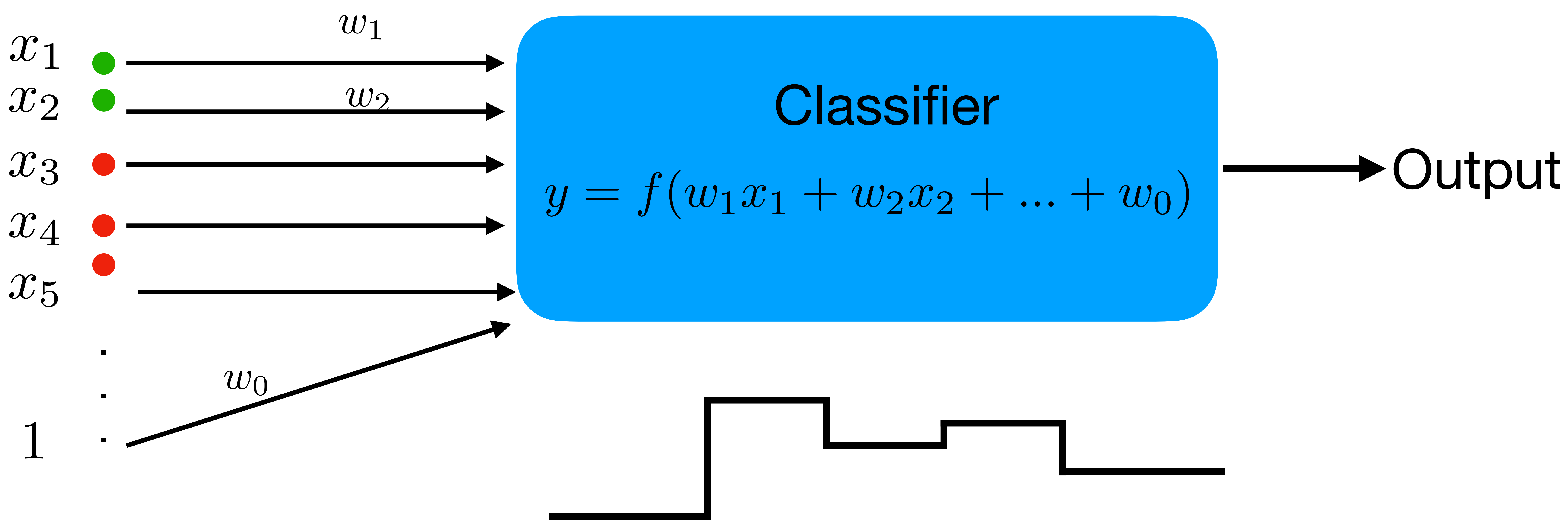


What features can produce this decision rule ?

Perceptron Classifier



Perceptron Classifier



Multi-layer Perceptron

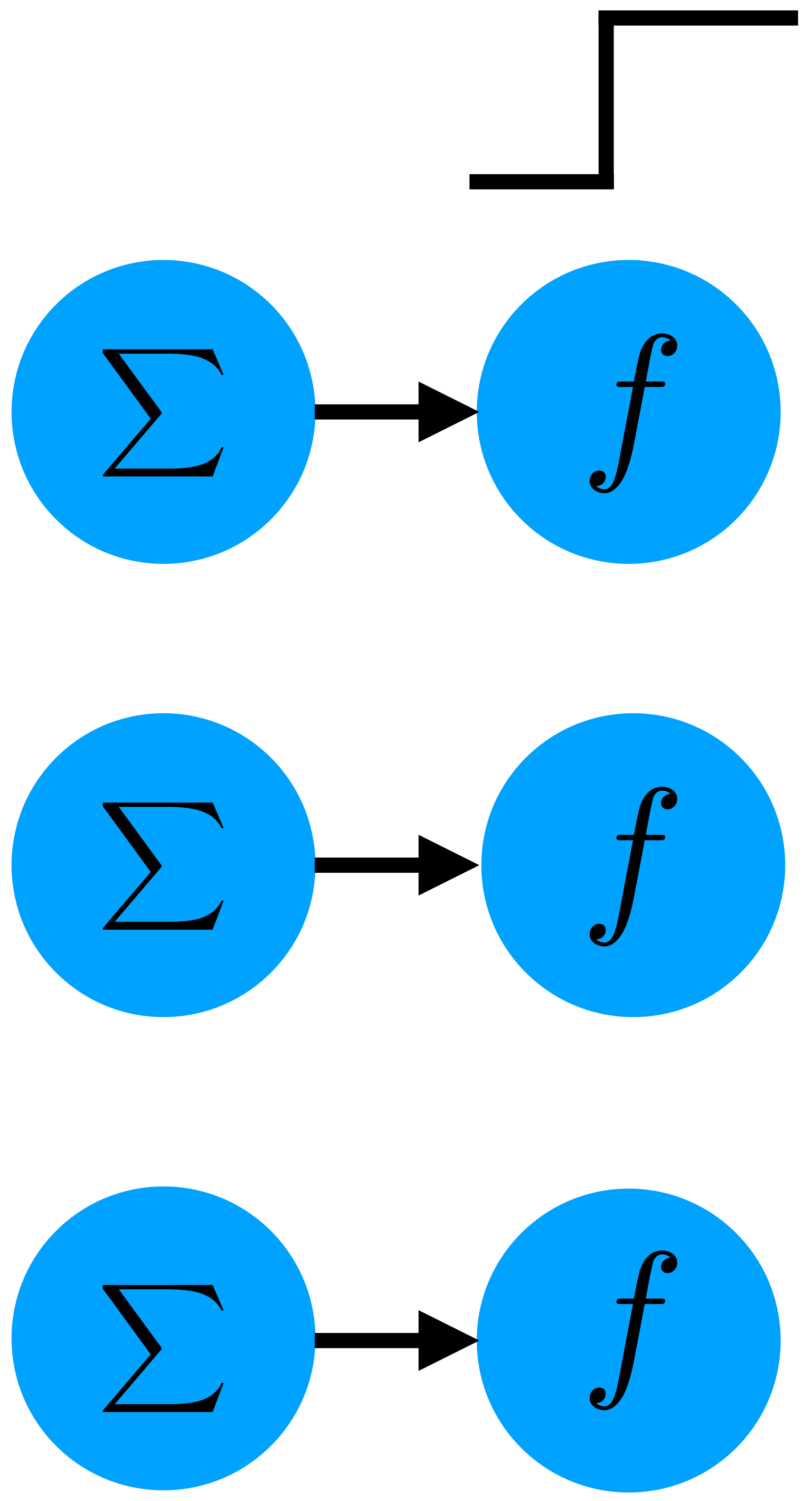
x_1

1

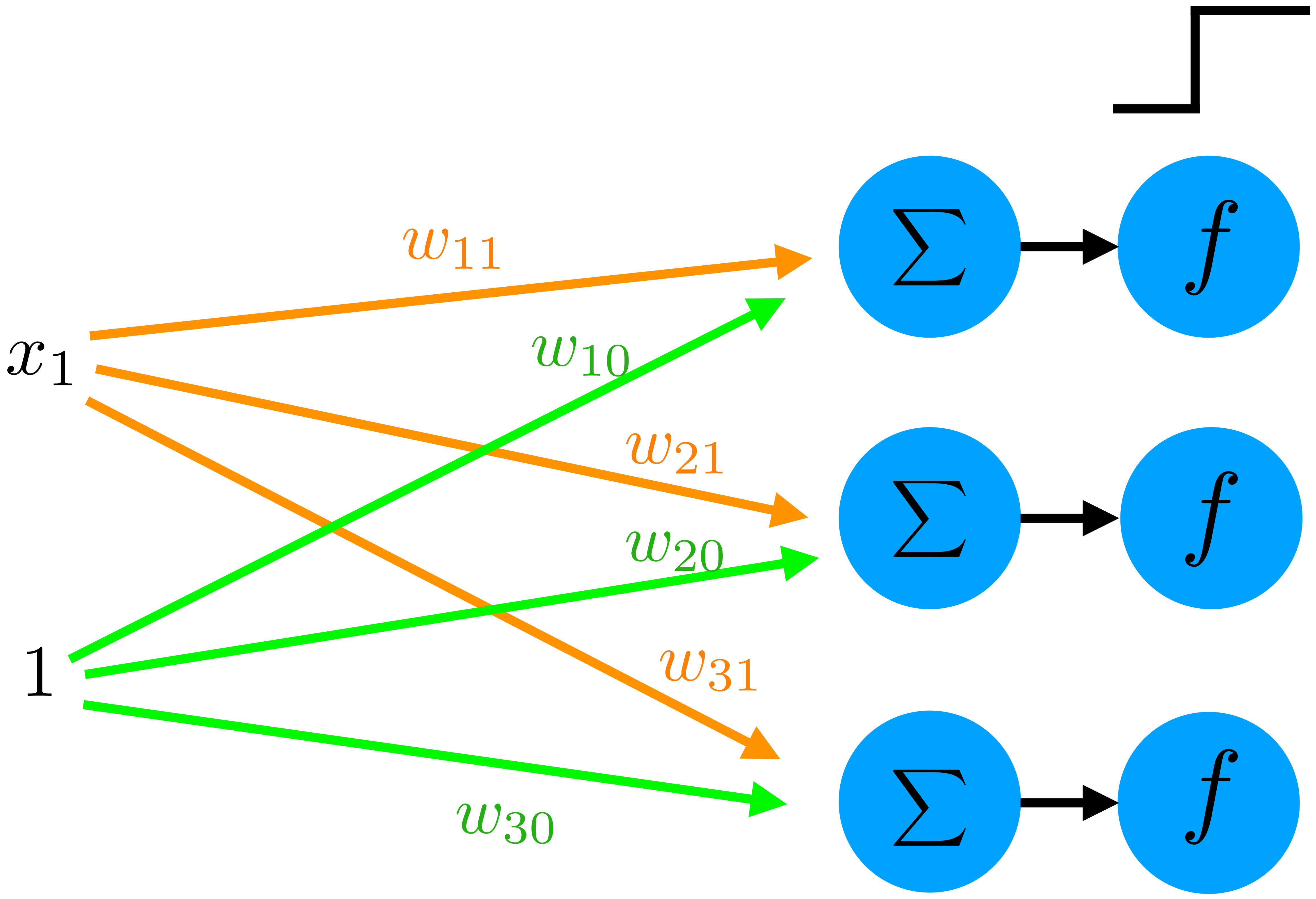
Multi-layer Perceptron

x_1

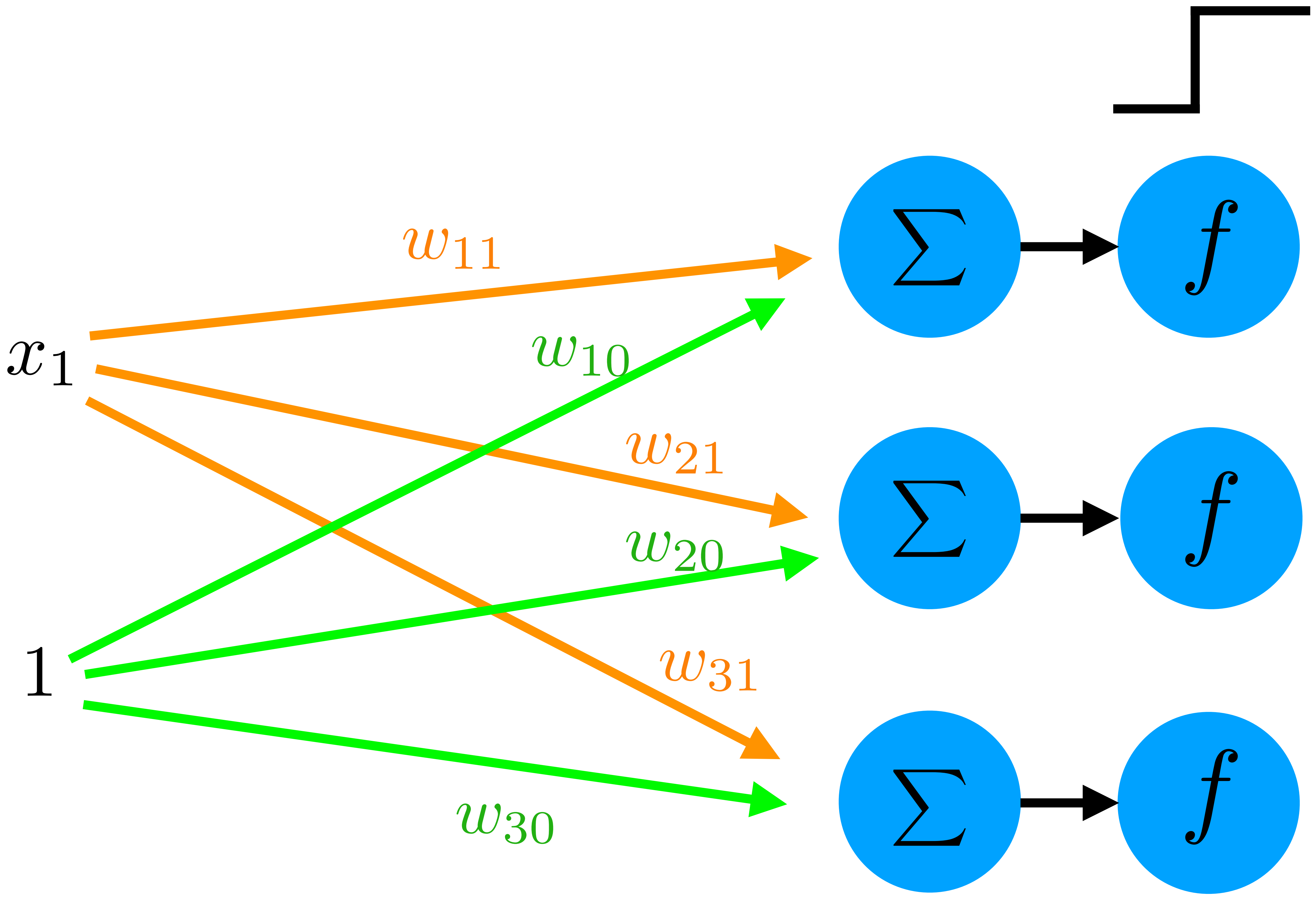
1



Multi-layer Perceptron



Multi-layer Perceptron

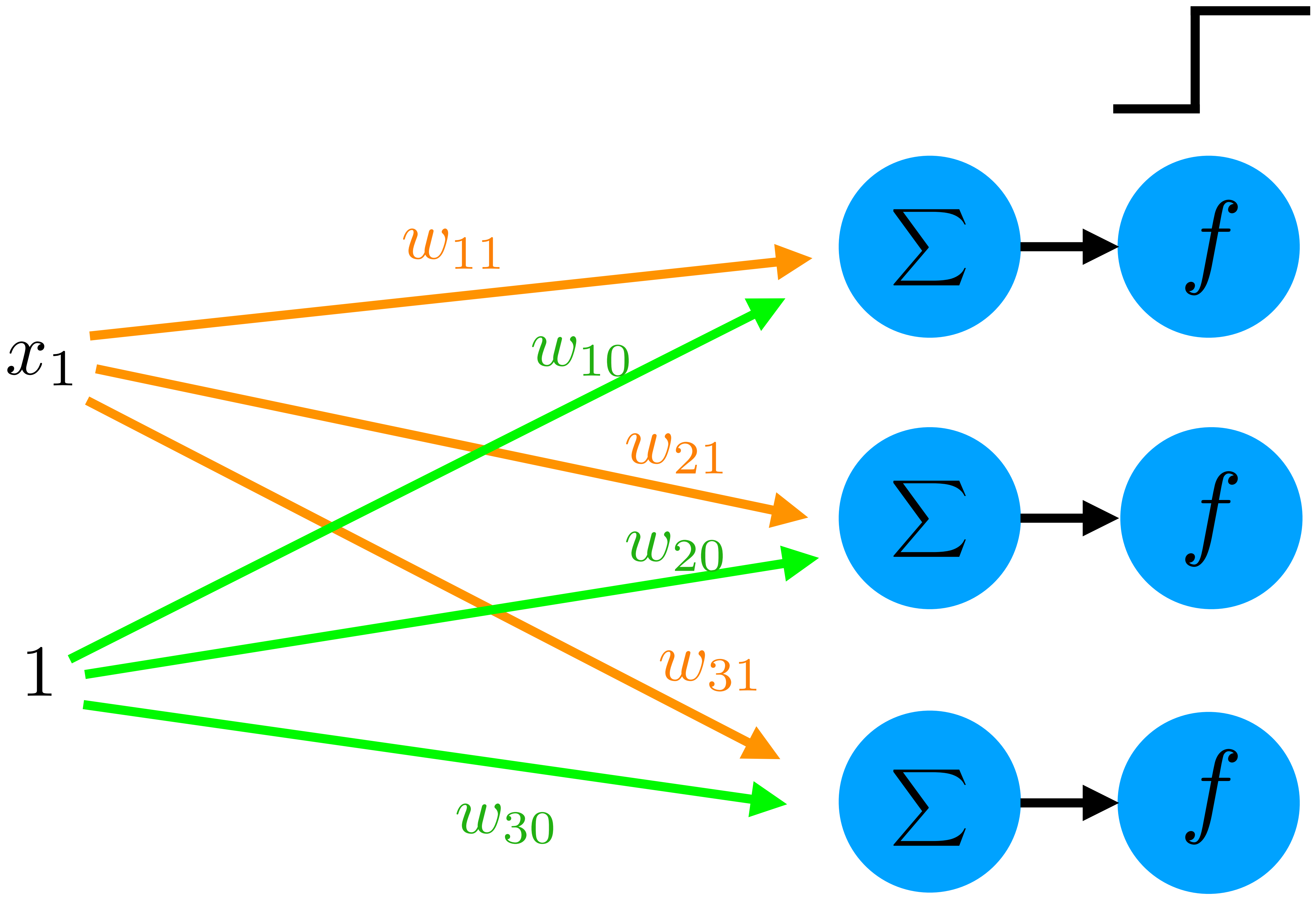


$x_1 \, w_{11} + w_{10}$

$x_1 \, w_{21} + w_{20}$

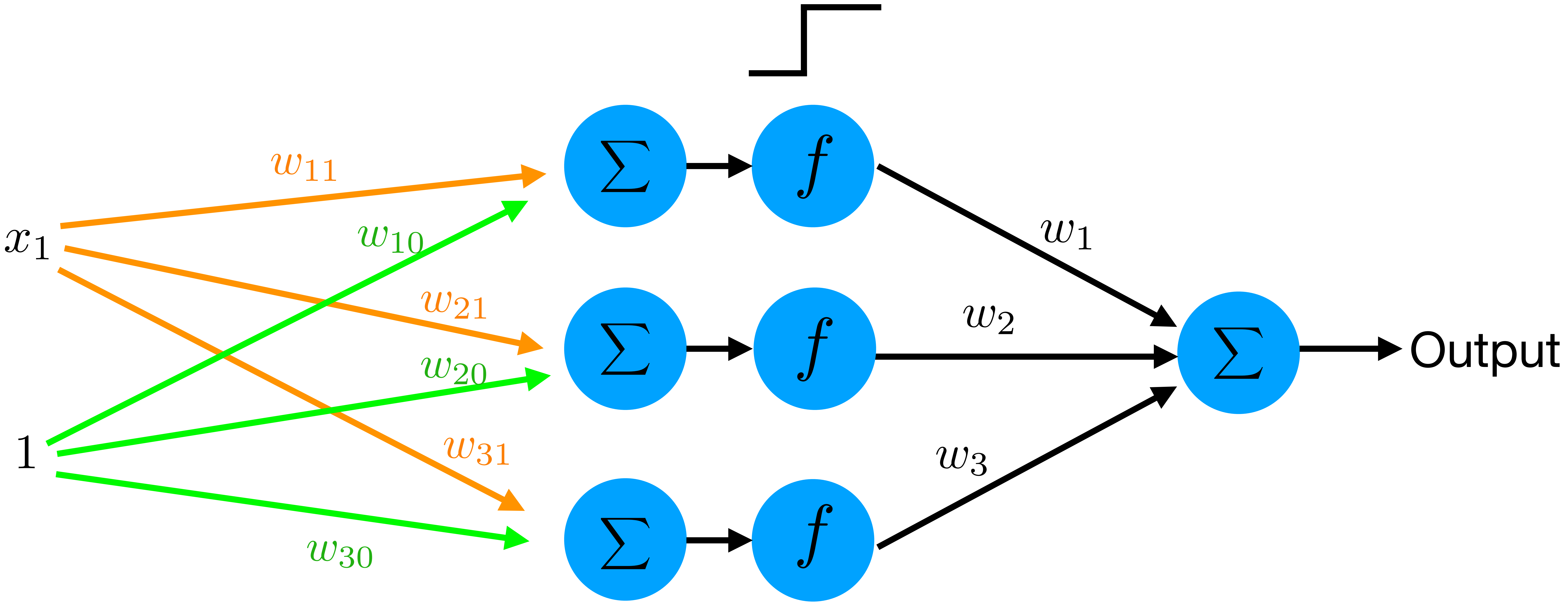
$x_1 \, w_{31} + w_{30}$

Multi-layer Perceptron



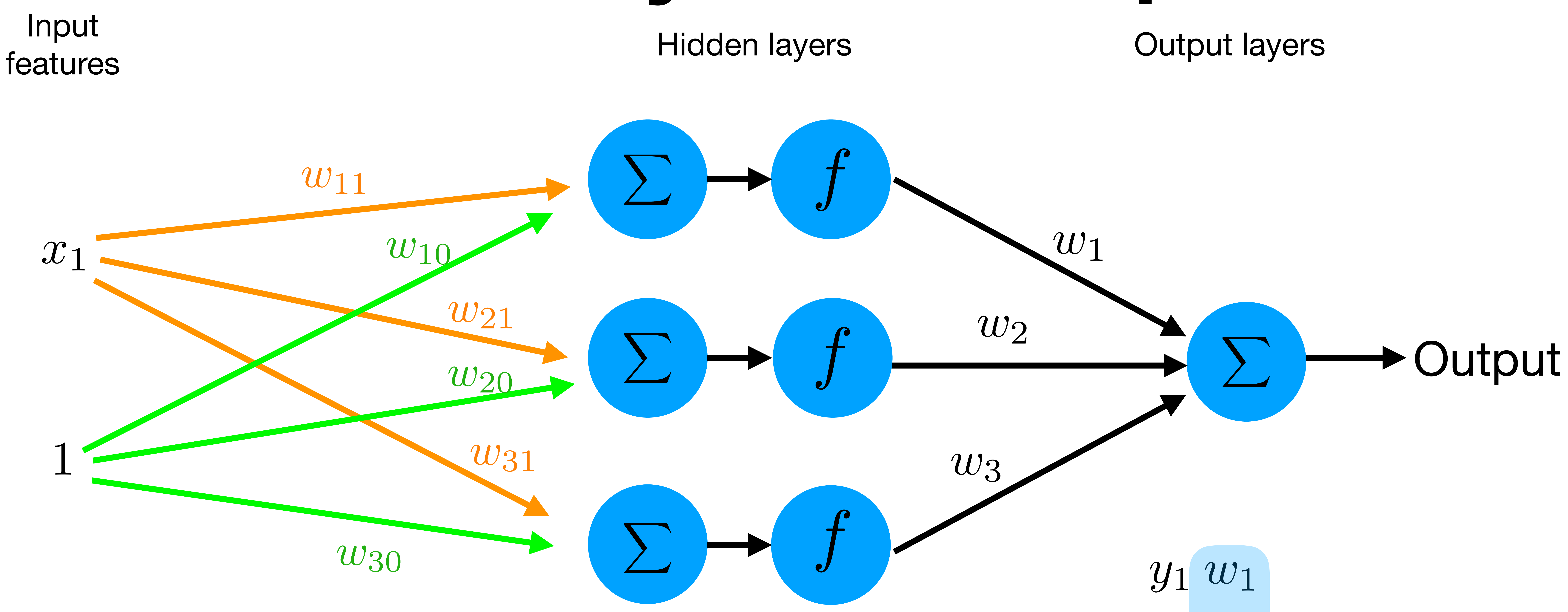
$$y_1 = f(x_1 w_{11} + w_{10})$$
$$y_2 = f(x_1 w_{21} + w_{20})$$
$$y_3 = f(x_1 w_{31} + w_{30})$$

Multi-layer Perceptron



$$y_1 = f(x_1 w_{11} + w_{10})$$
$$y_2 = f(x_1 w_{21} + w_{20})$$
$$y_3 = f(x_1 w_{31} + w_{30})$$

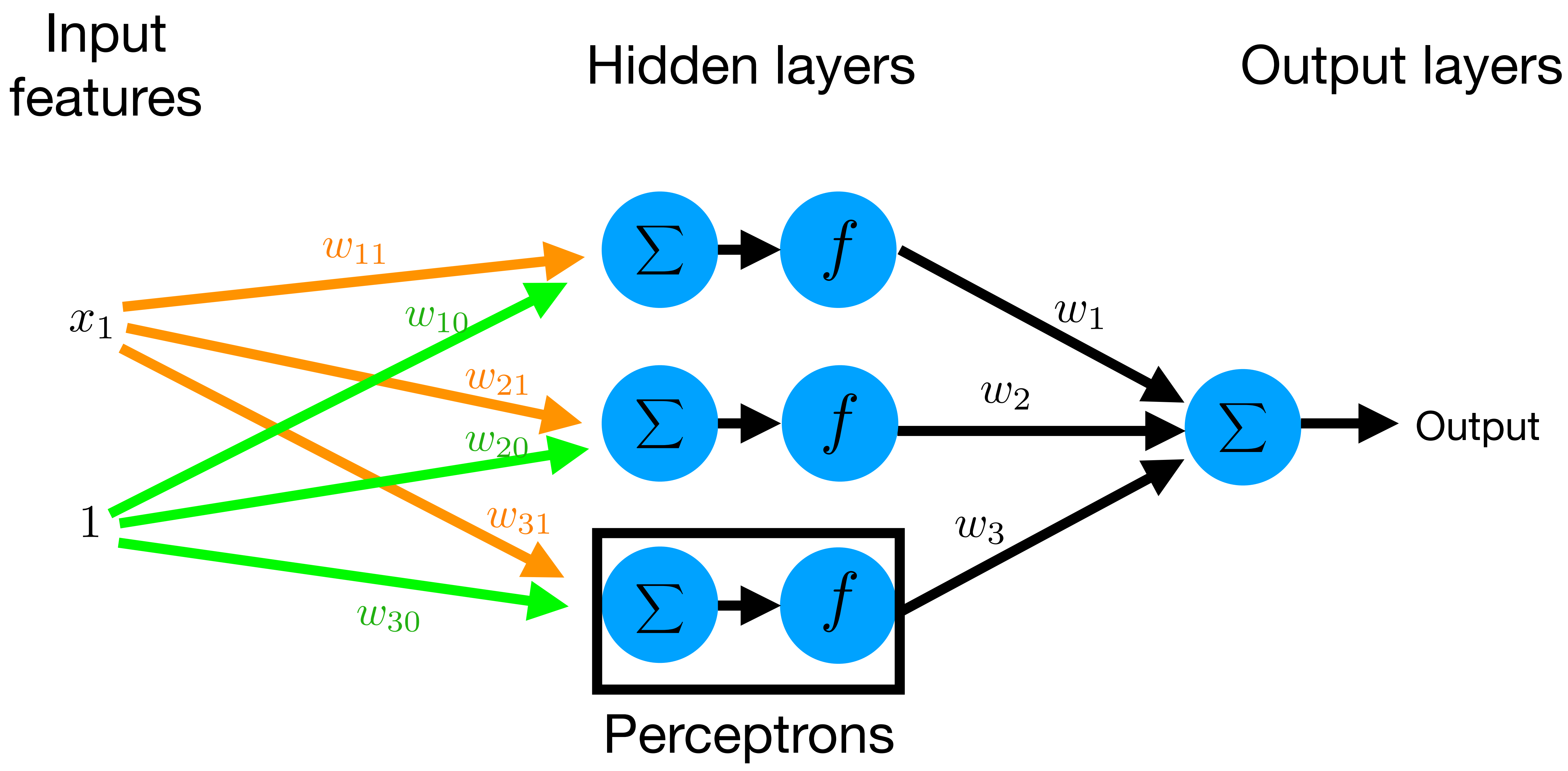
Multi-layer Perceptron



$y_1 = f(x_1 w_{11} + w_{10})$
$y_2 = f(x_1 w_{21} + w_{20})$
$y_3 = f(x_1 w_{31} + w_{30})$

$y_1 \; w_1$
$y_2 \; w_2$
$y_3 \; w_3$

Multi-layer Perceptron



"Features" are outputs of perceptrons

Matrix of first layer weights

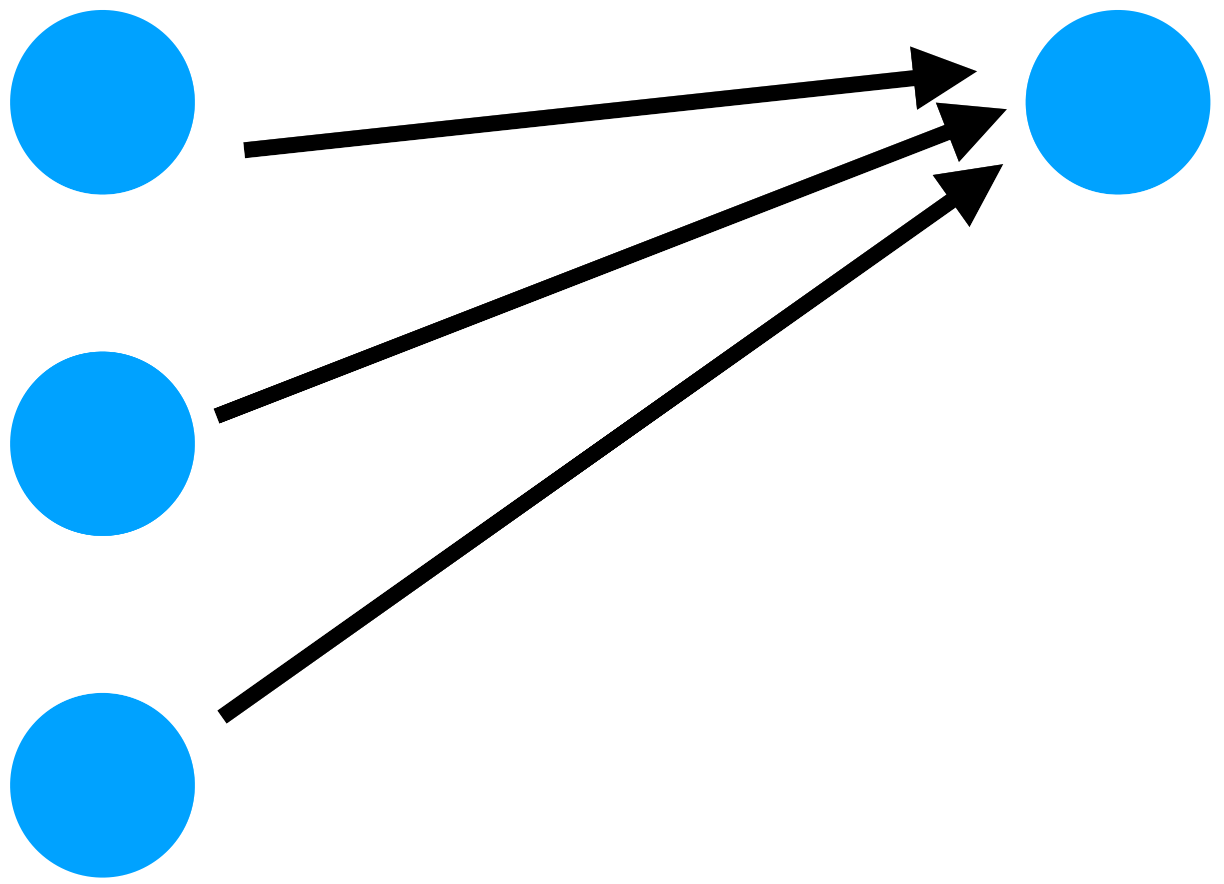
$$\begin{matrix} w_{11} & w_{10} \\ w_{21} & w_{20} \\ w_{31} & w_{30} \end{matrix}$$

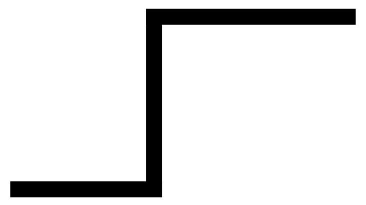
Matrix of second layer weights

$$\begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix}$$

Features of MLPs

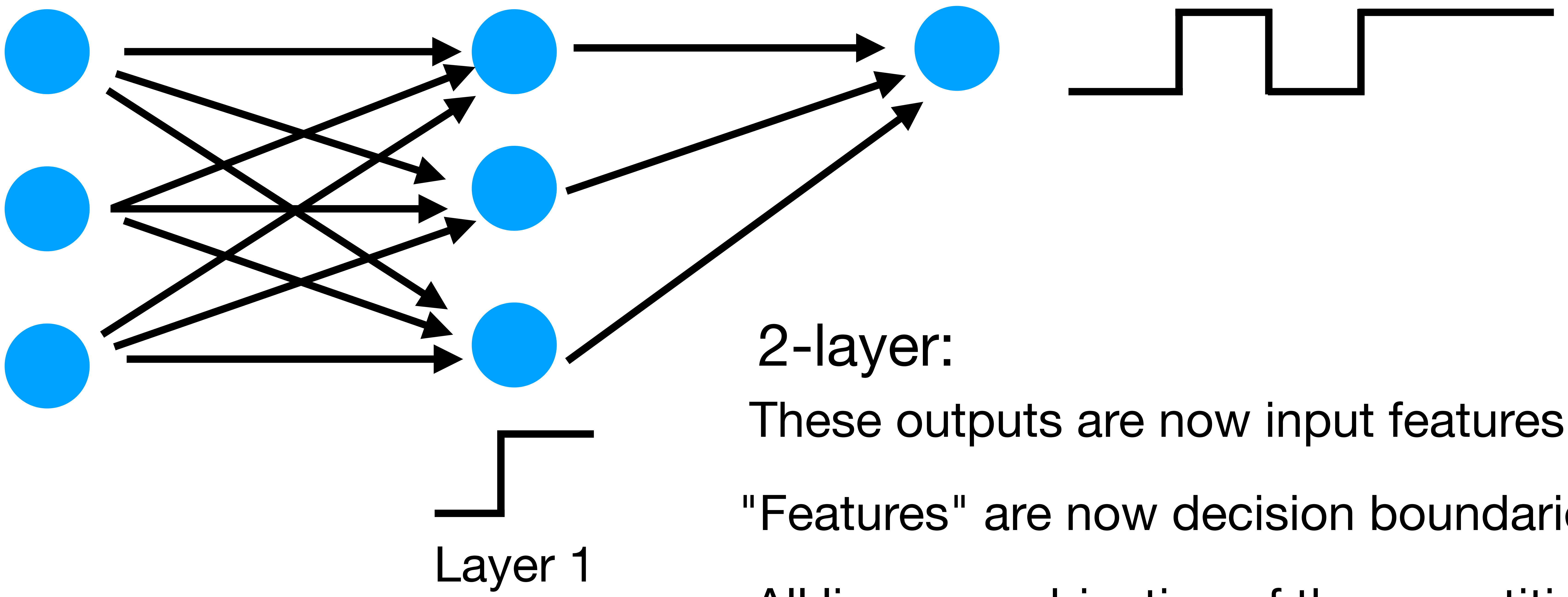
Input
features





Perceptron: Step function
with linear decision boundary

Features of MLPs



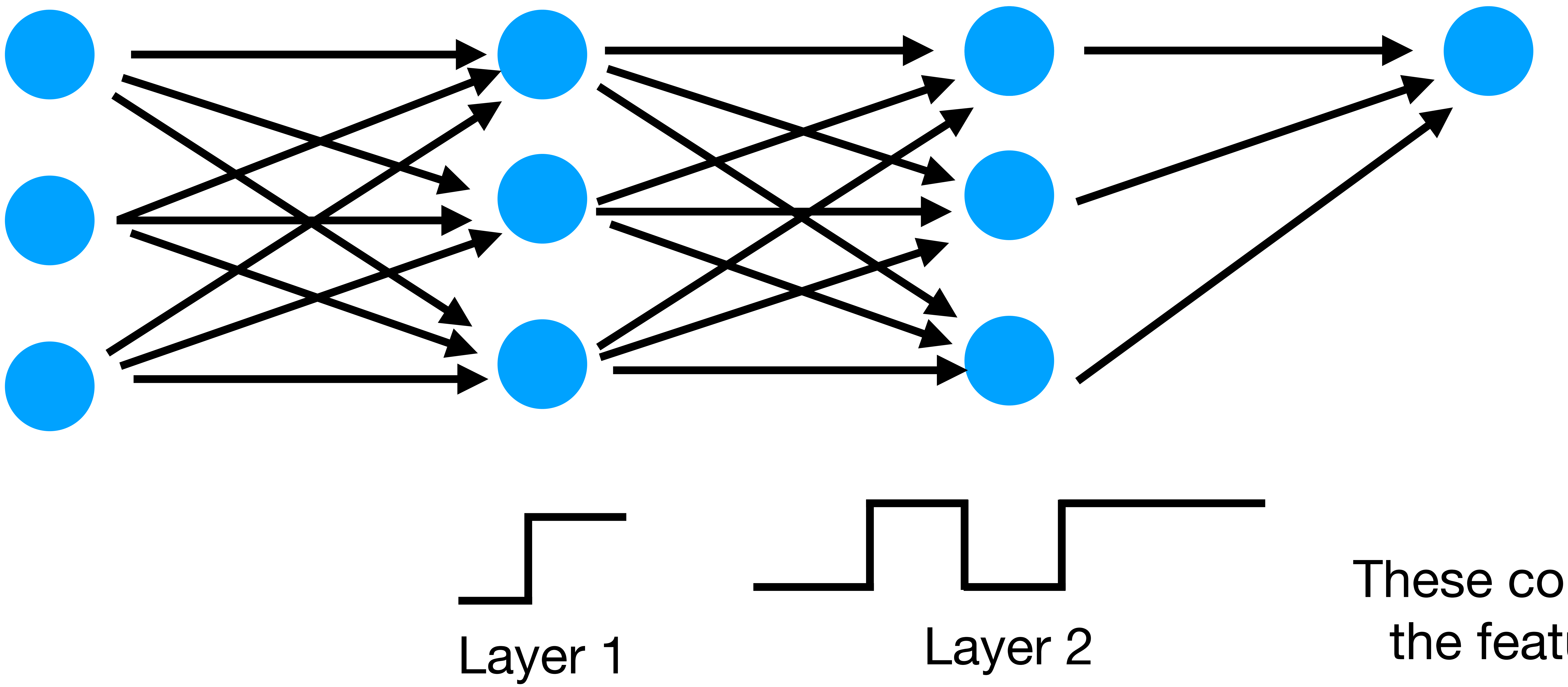
2-layer:

These outputs are now input features to the next layer

"Features" are now decision boundaries (partitions)

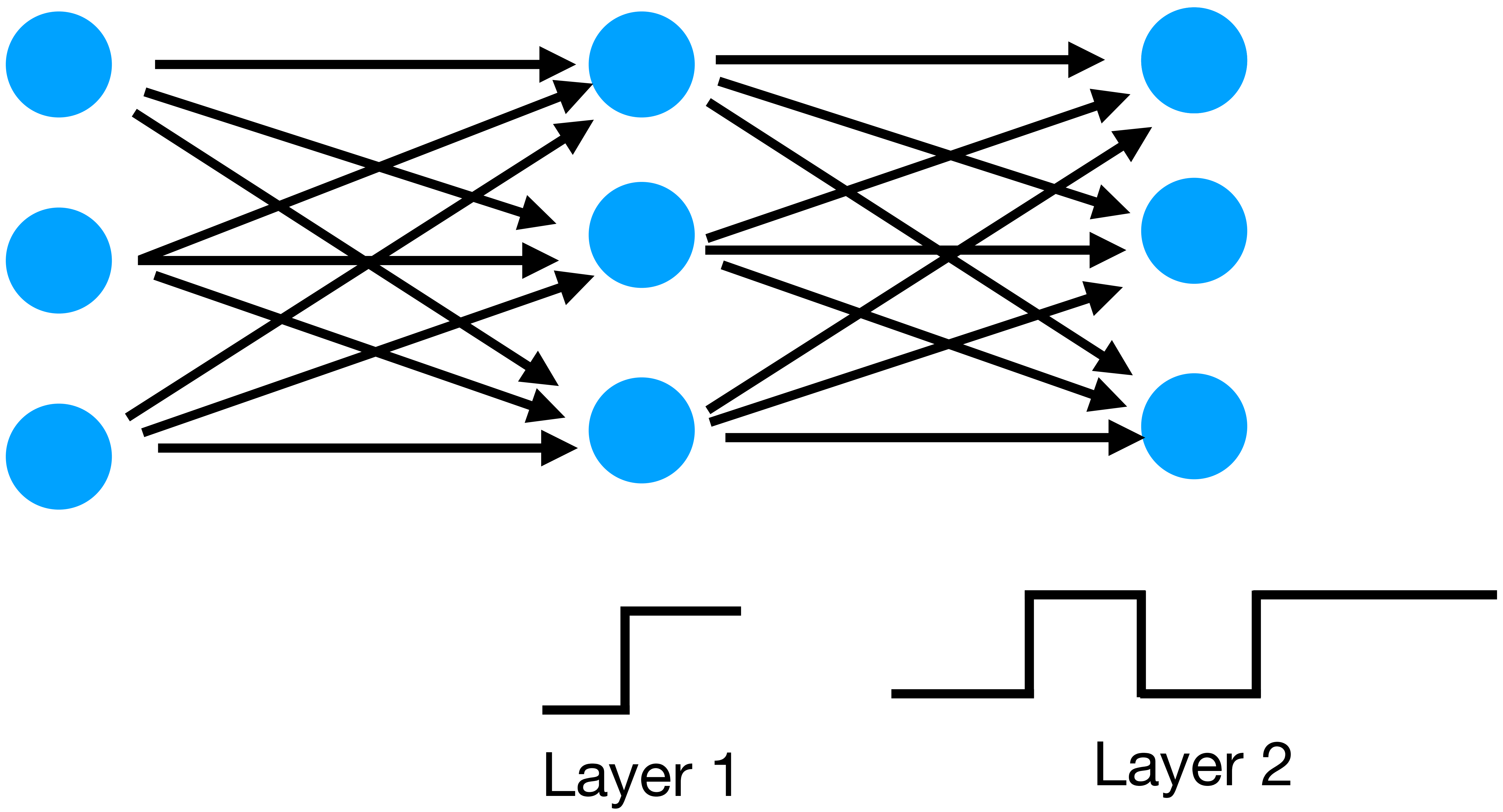
All linear combination of those partitions give complex partitions

Features of MLPs



These complex outputs become the features for the new layer

Features of MLPs



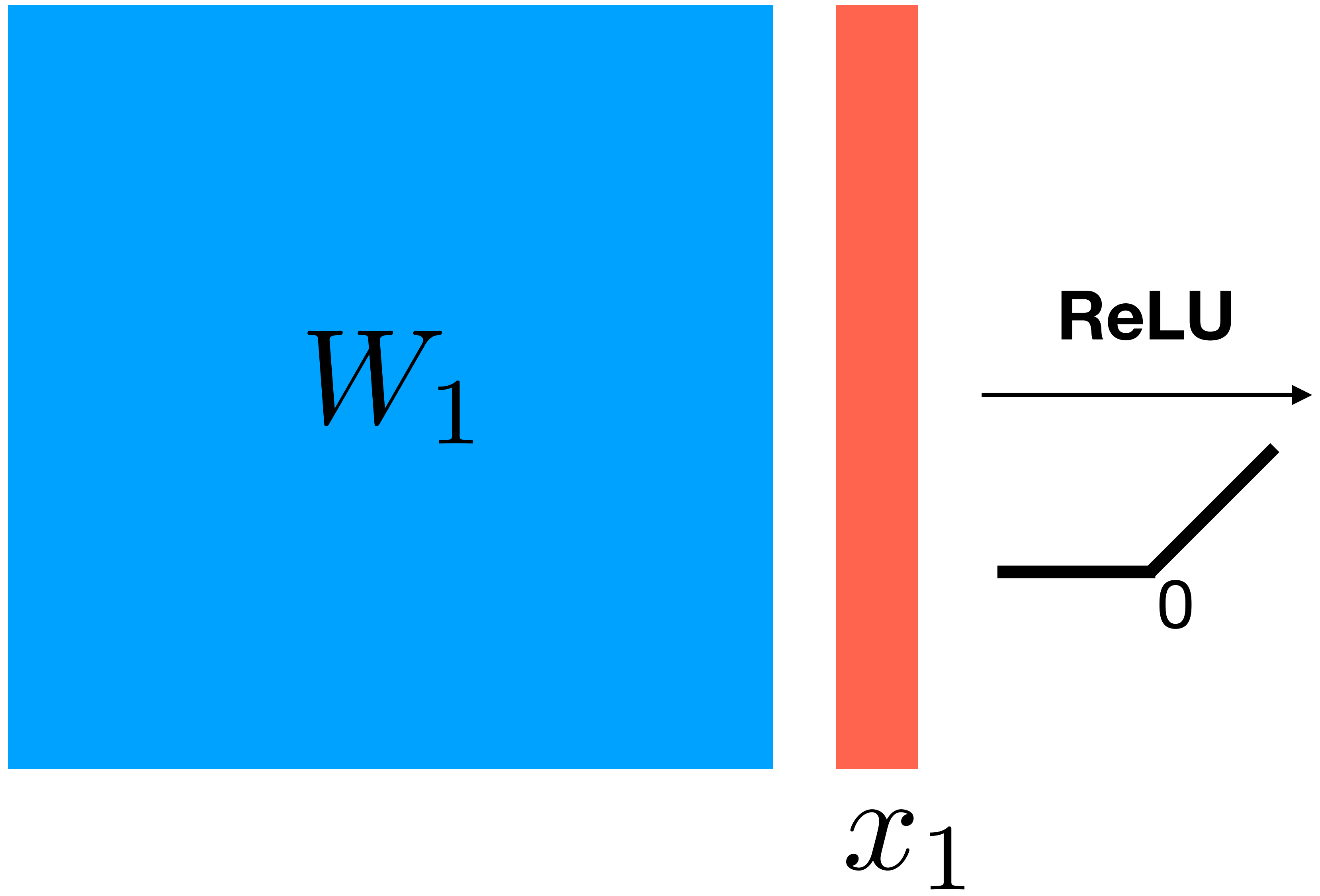
...

Deep Neural Networks

Computational Graph representation of Neural Networks

Neural Networks

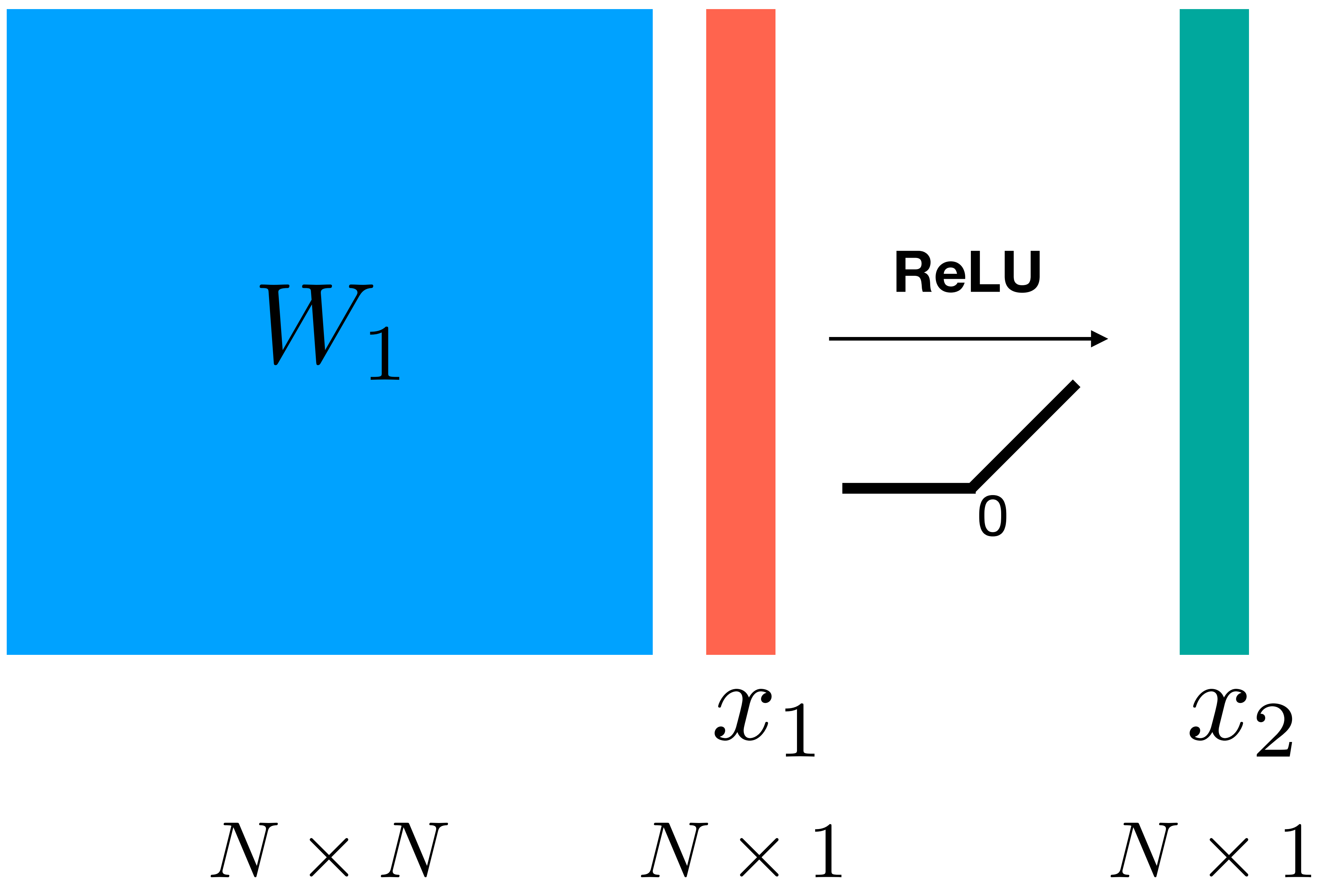
Fully connected layers



$N \times N$ $\quad$ $N \times 1$

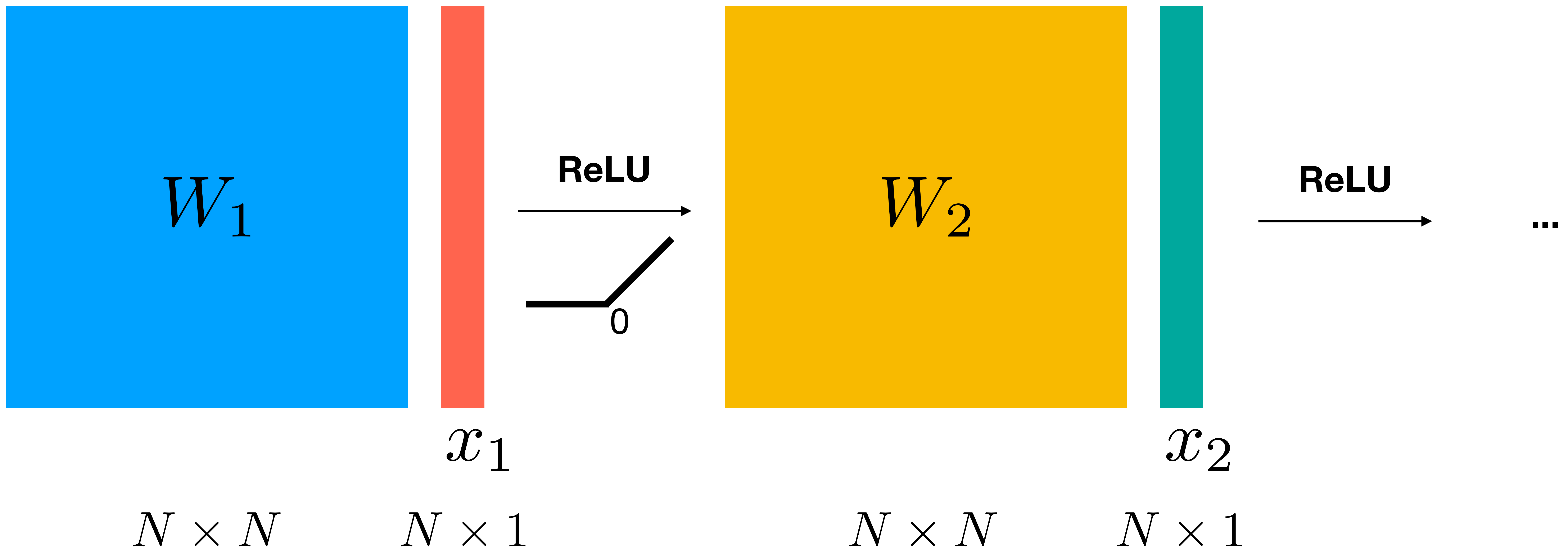
Neural Networks

Fully connected layers

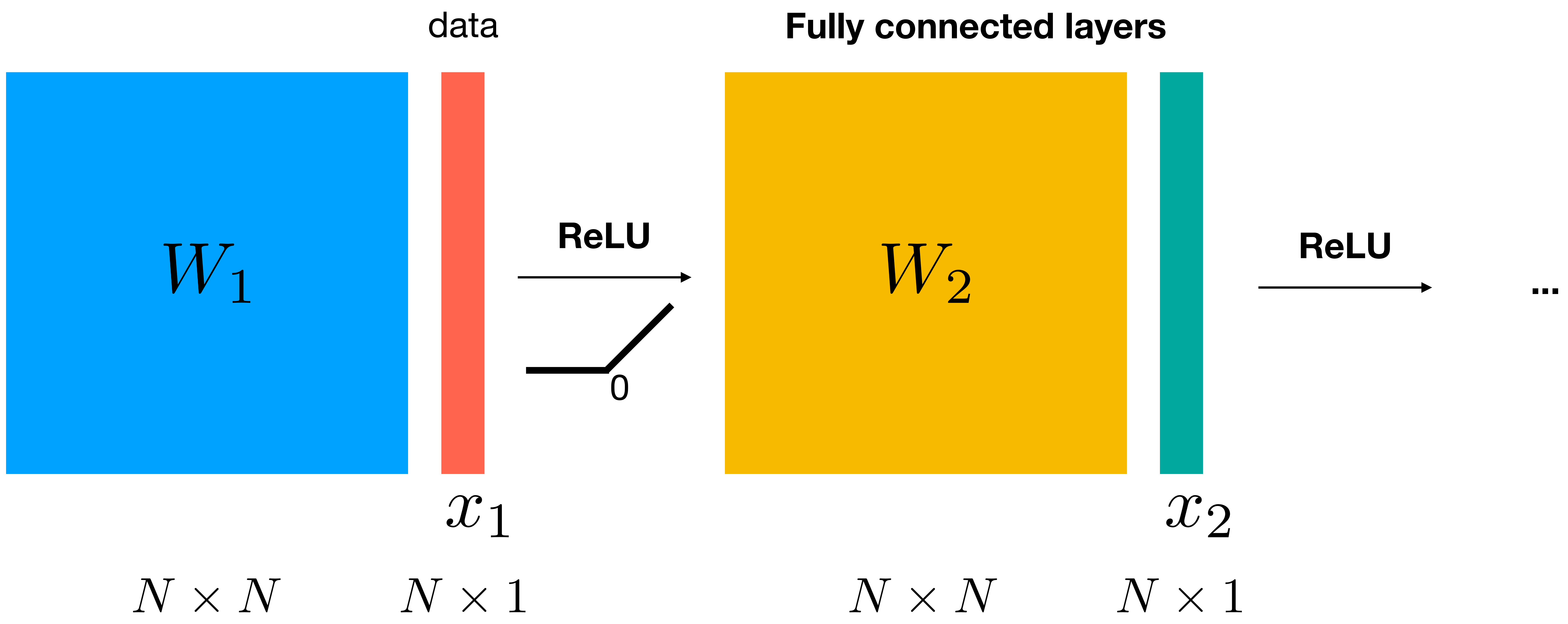


Neural Networks

Fully connected layers

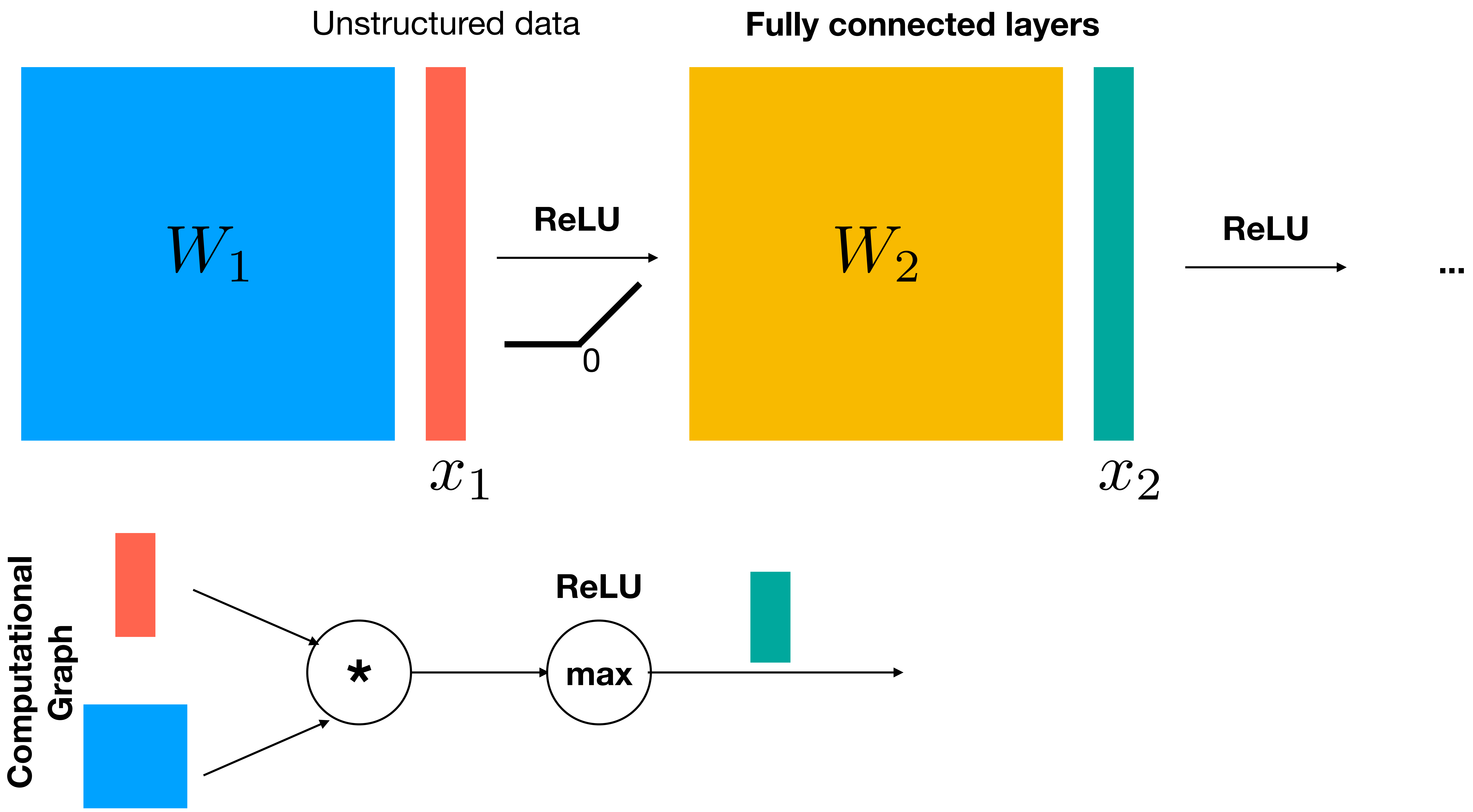


Neural Networks

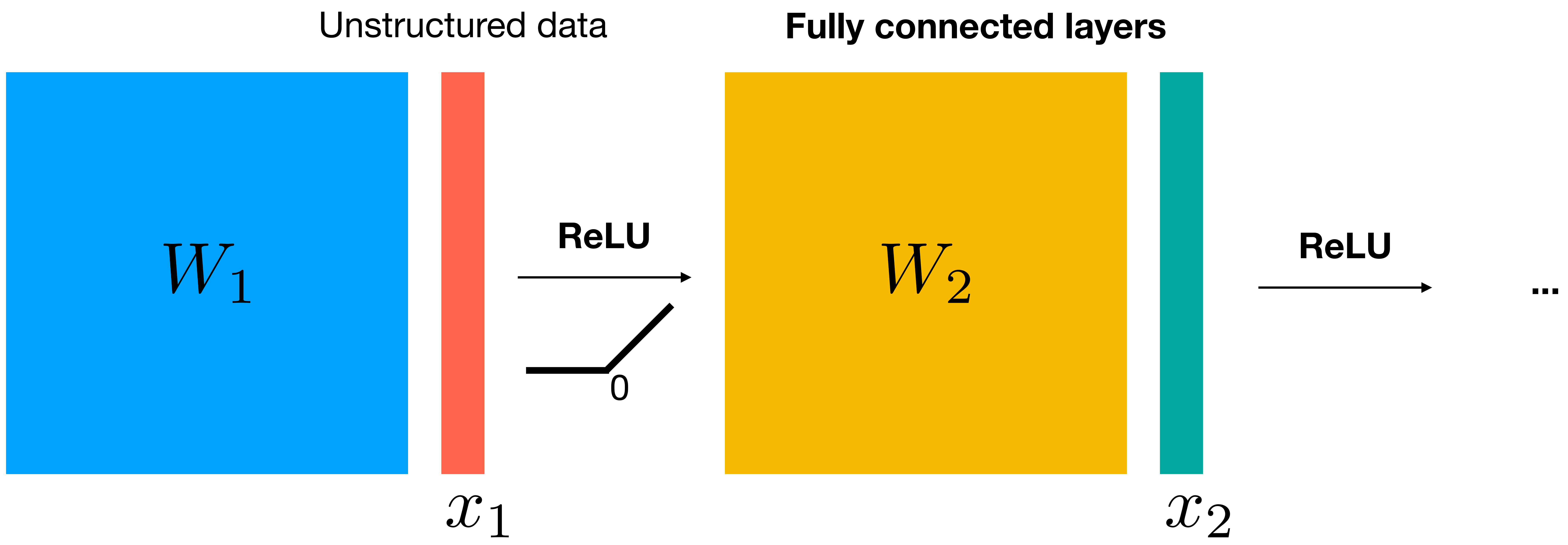


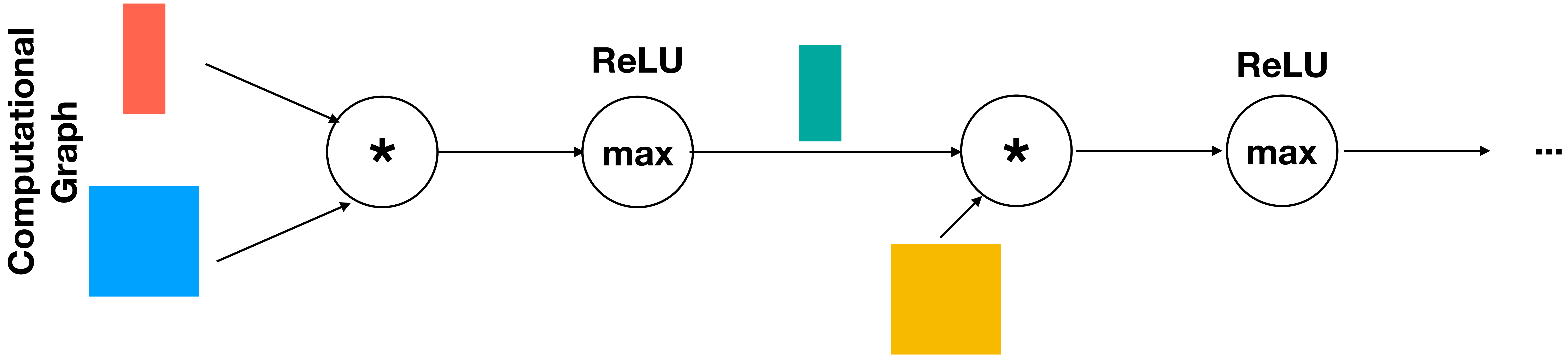
N represents number of pixels in an image

Neural Networks

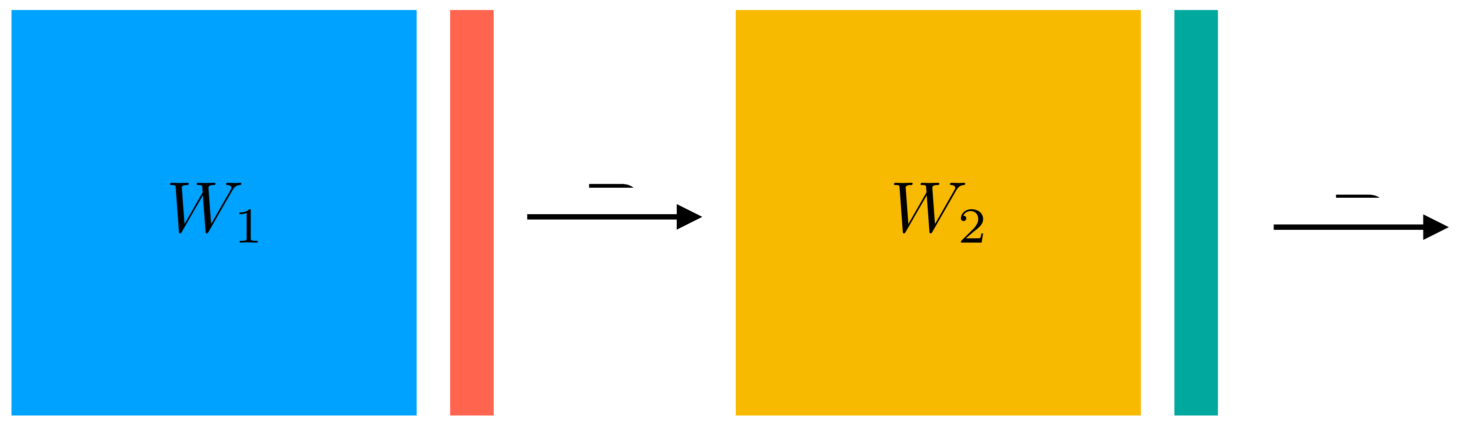


Neural Networks

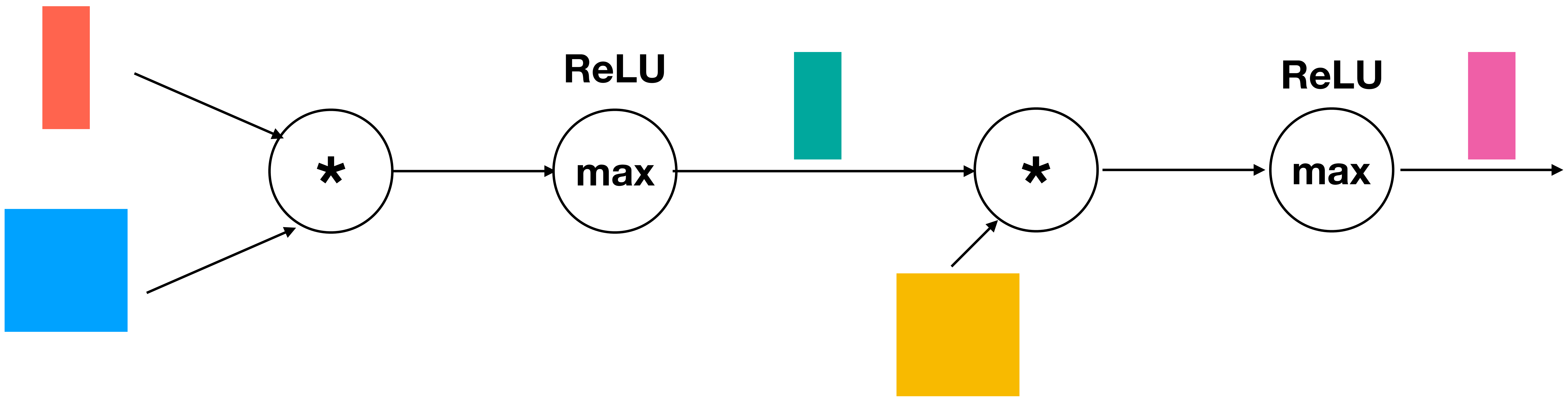




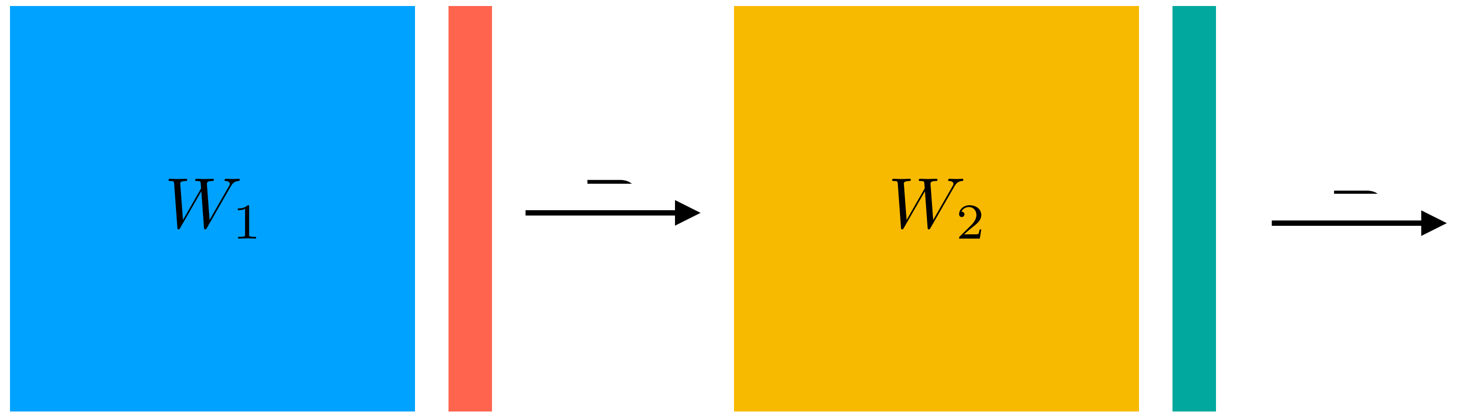
Two-layer model



Fully connected layers

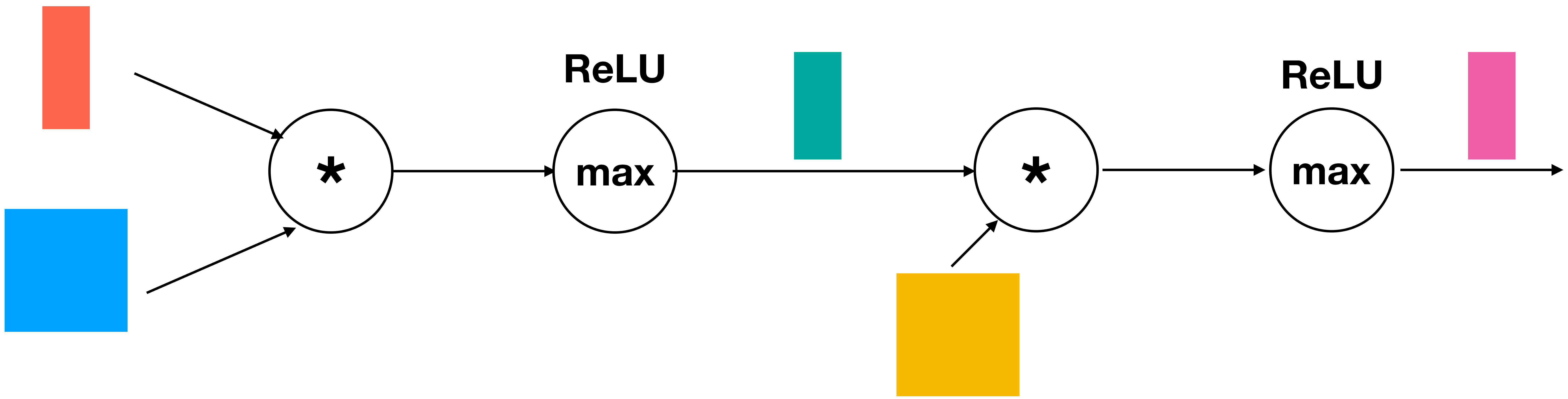


What can be a loss function ?



Two-layer model

Fully connected layers

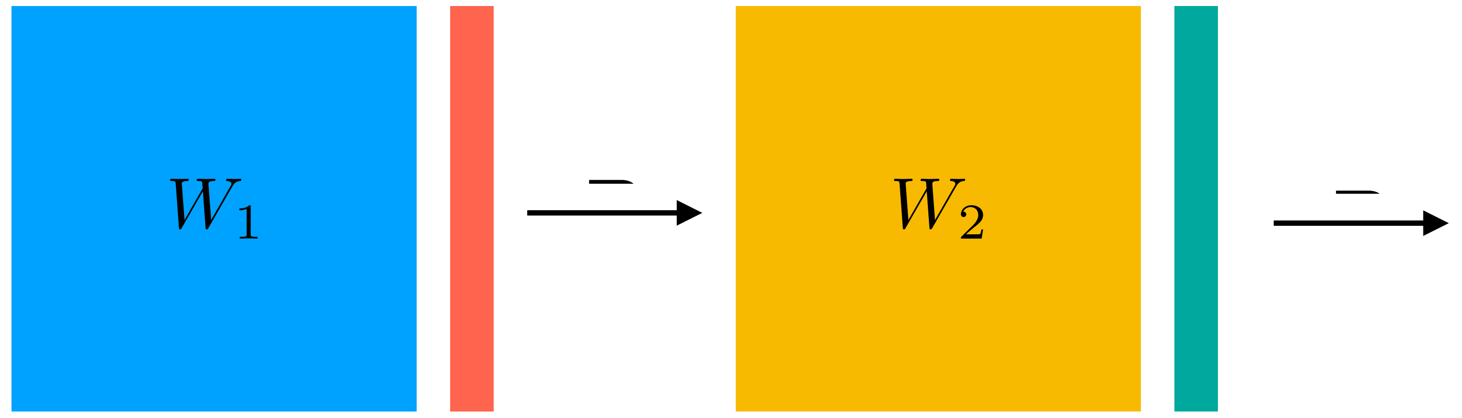


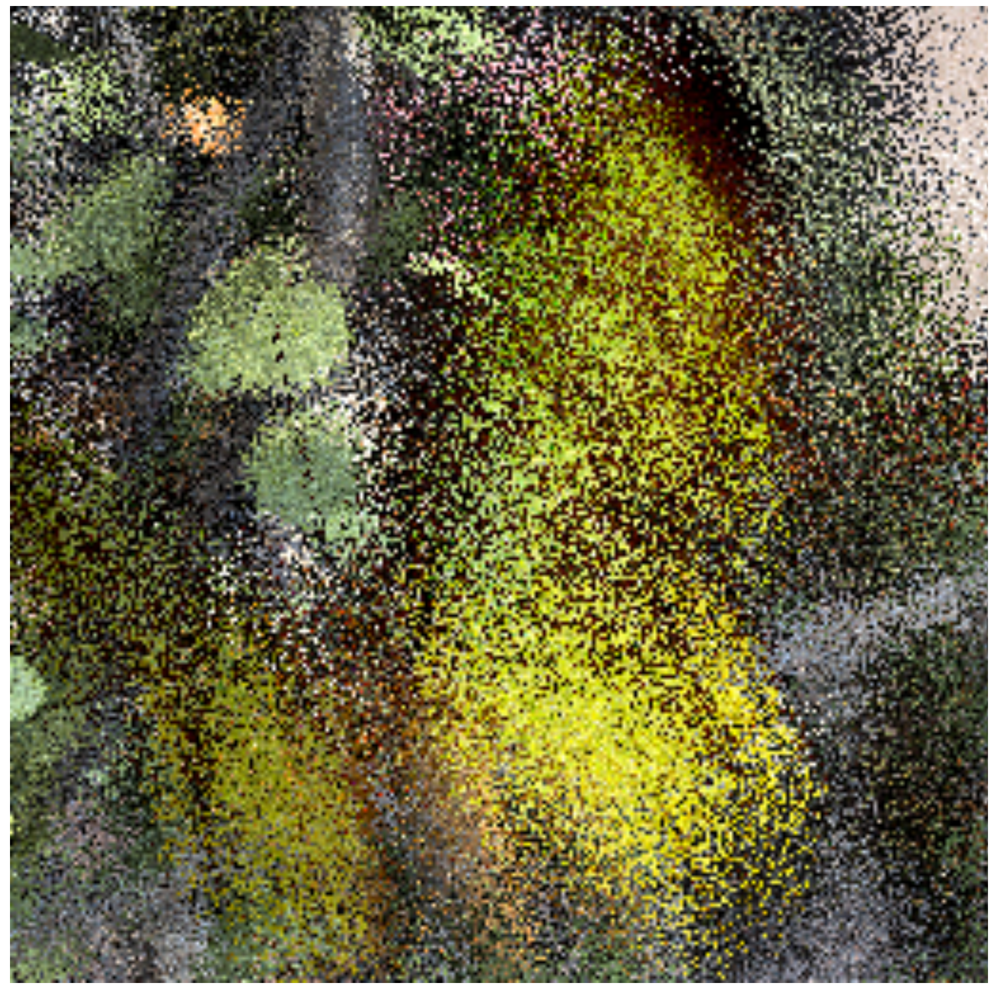
Reference

What can be a loss function ?

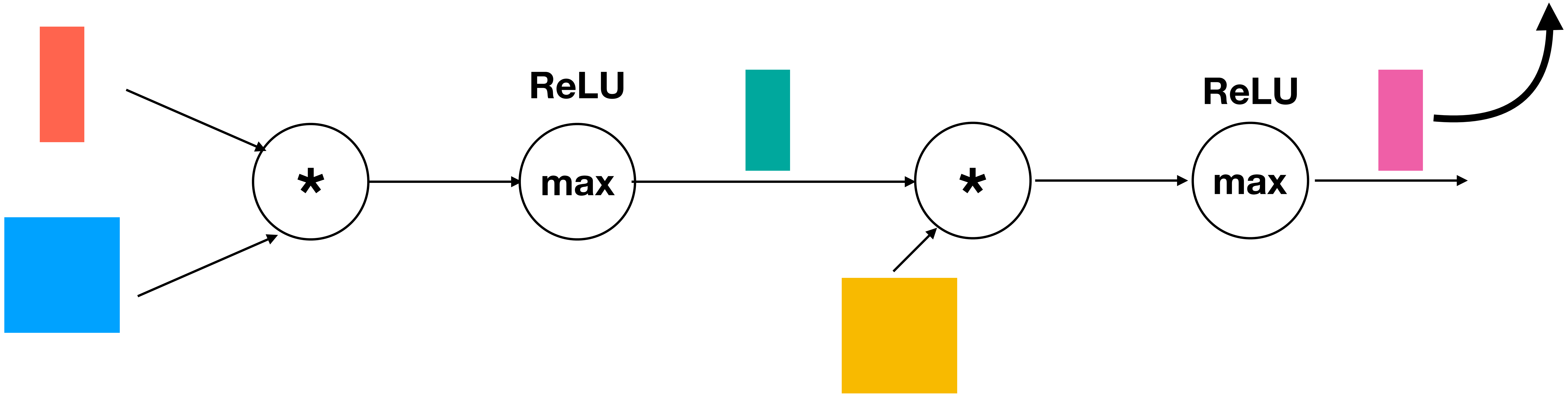
Two-layer model

Fully connected layers





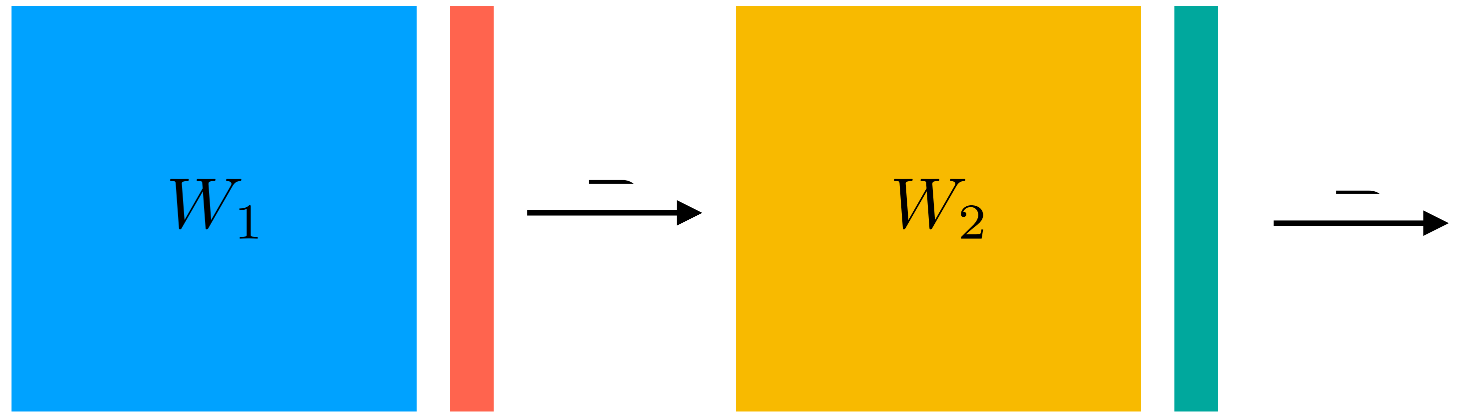
Reference

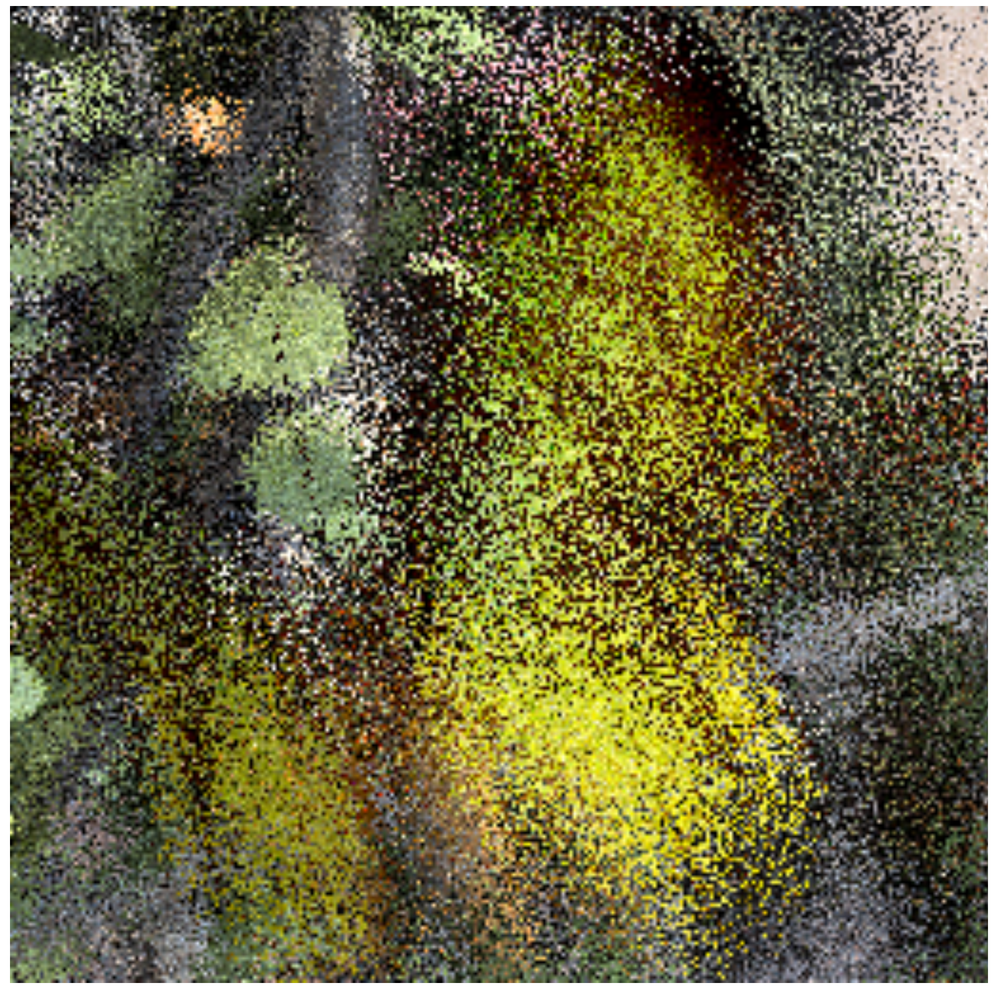


What can be a loss function ?

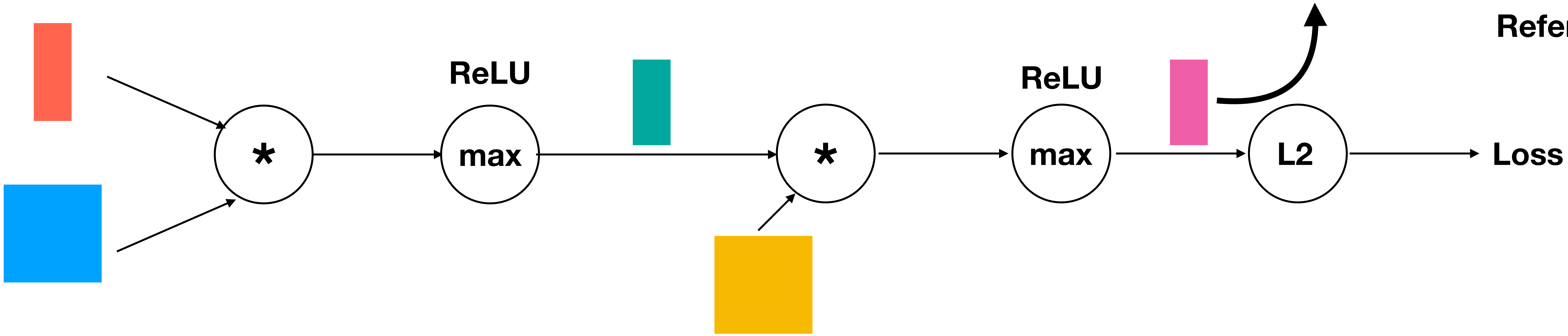
Two-layer model

Fully connected layers



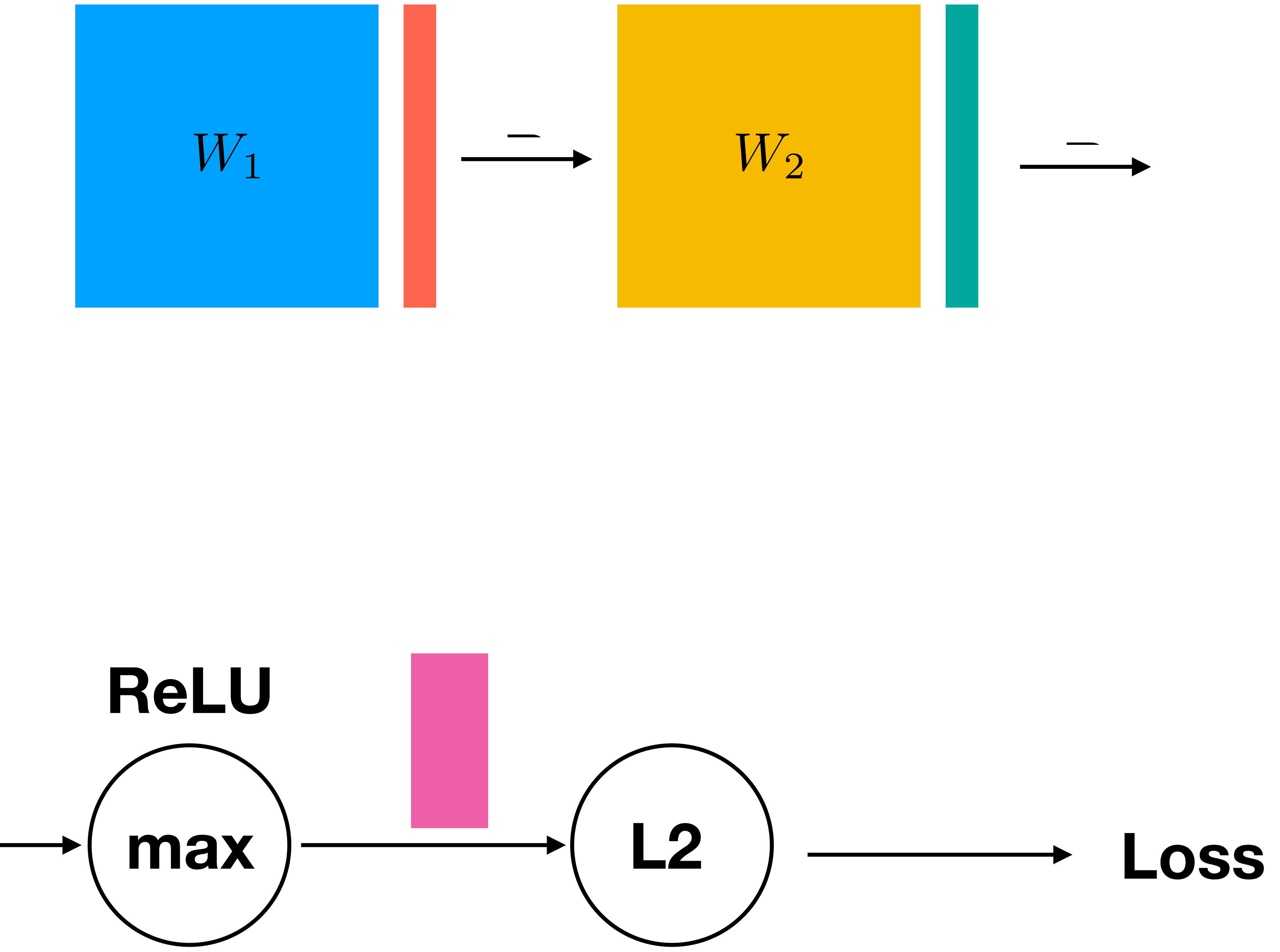


Reference



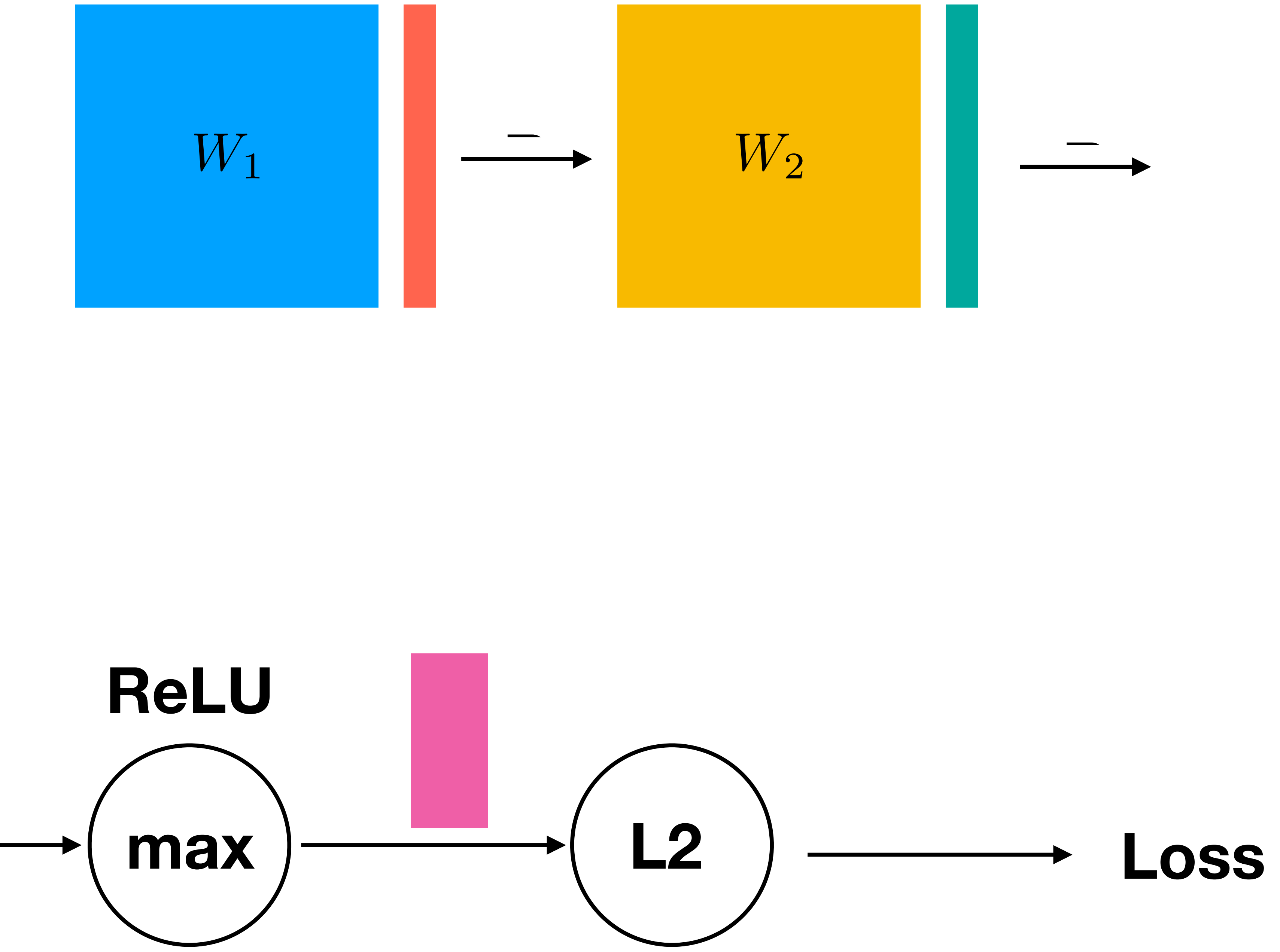
What can be a loss function ?

Two-layer model



What can be a loss function ?

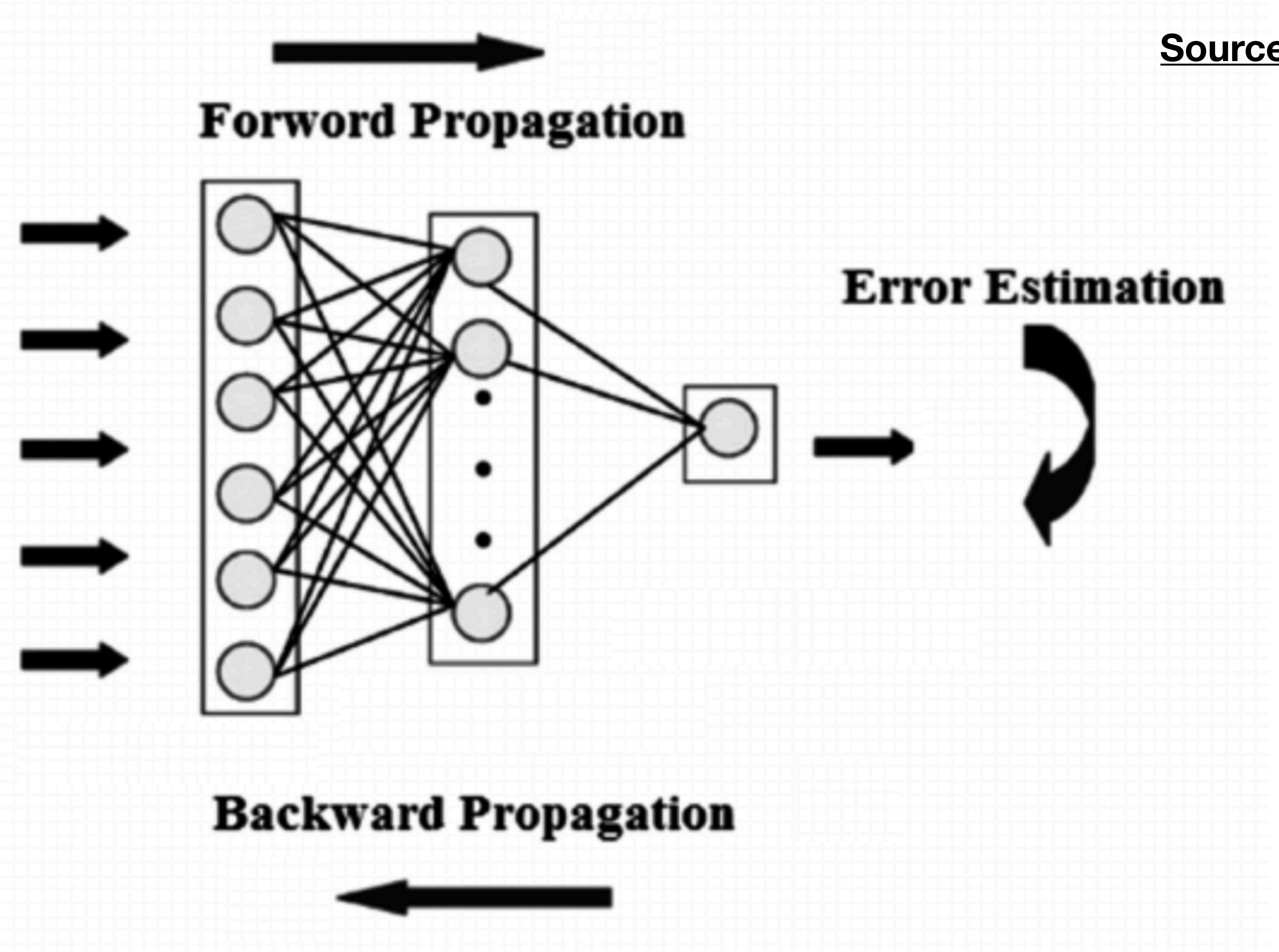
Two-layer model



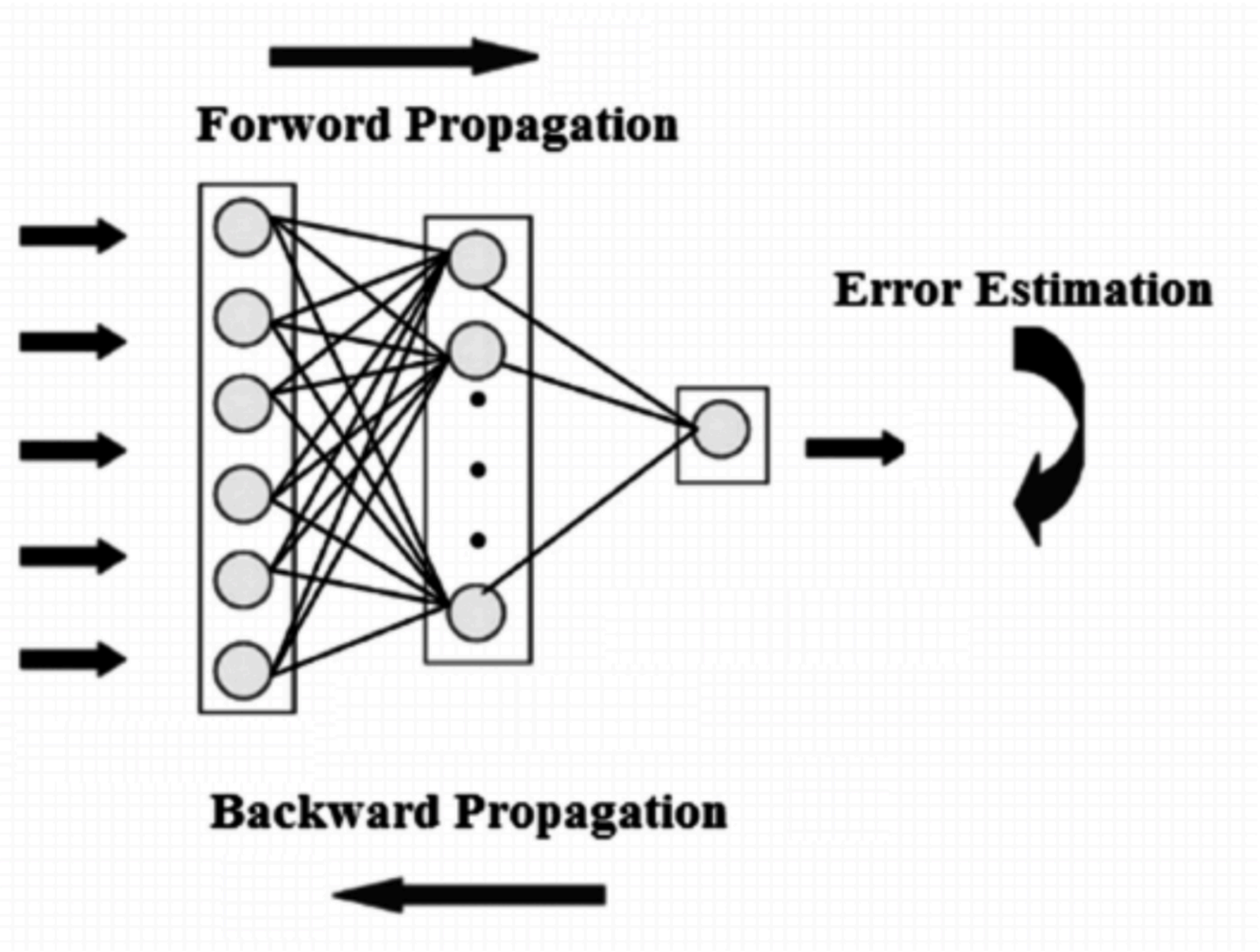
What can be a loss function ?

Two-layer model: Back propagation

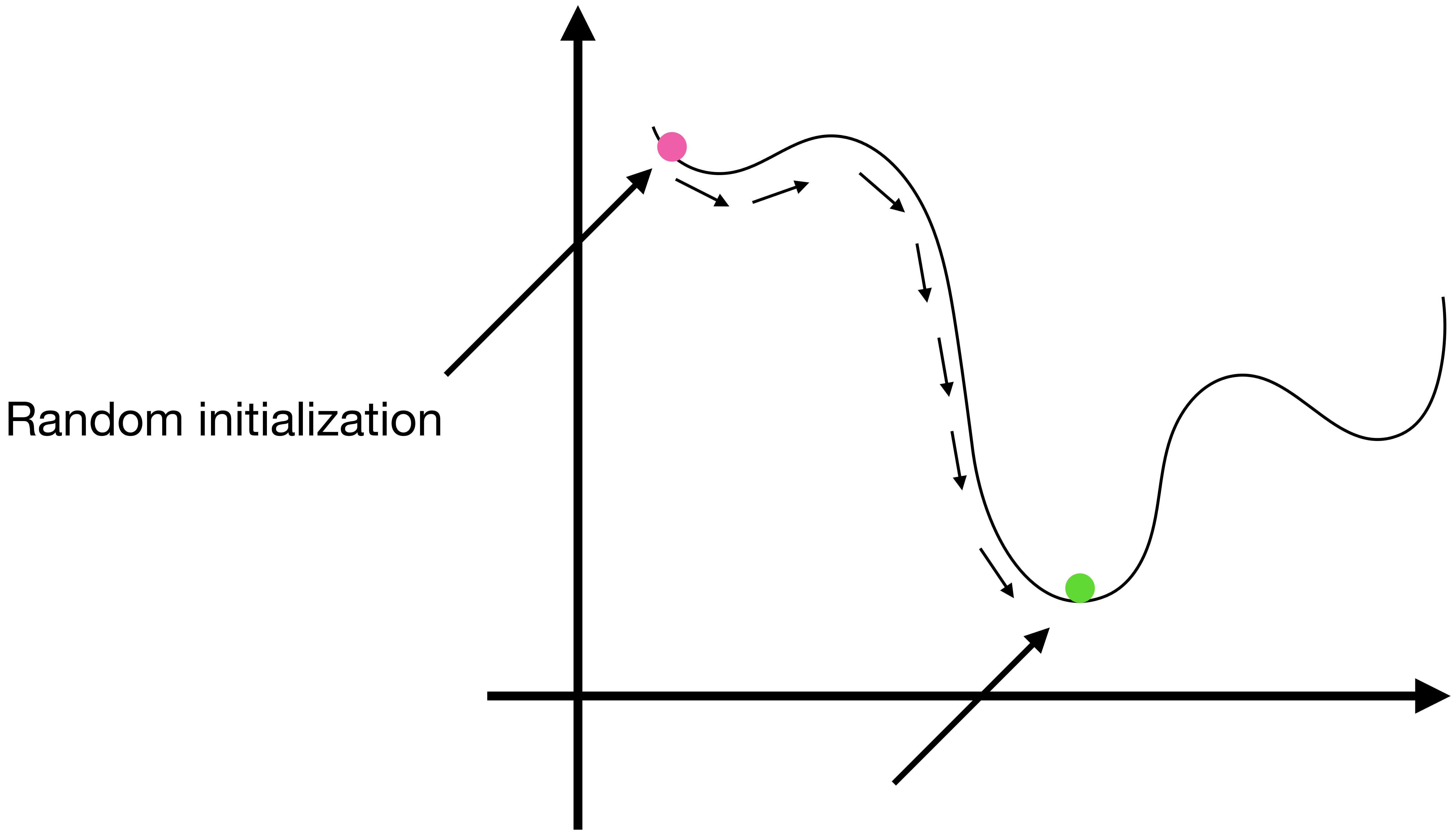
[Source link]



Two-layer model: Back propagation

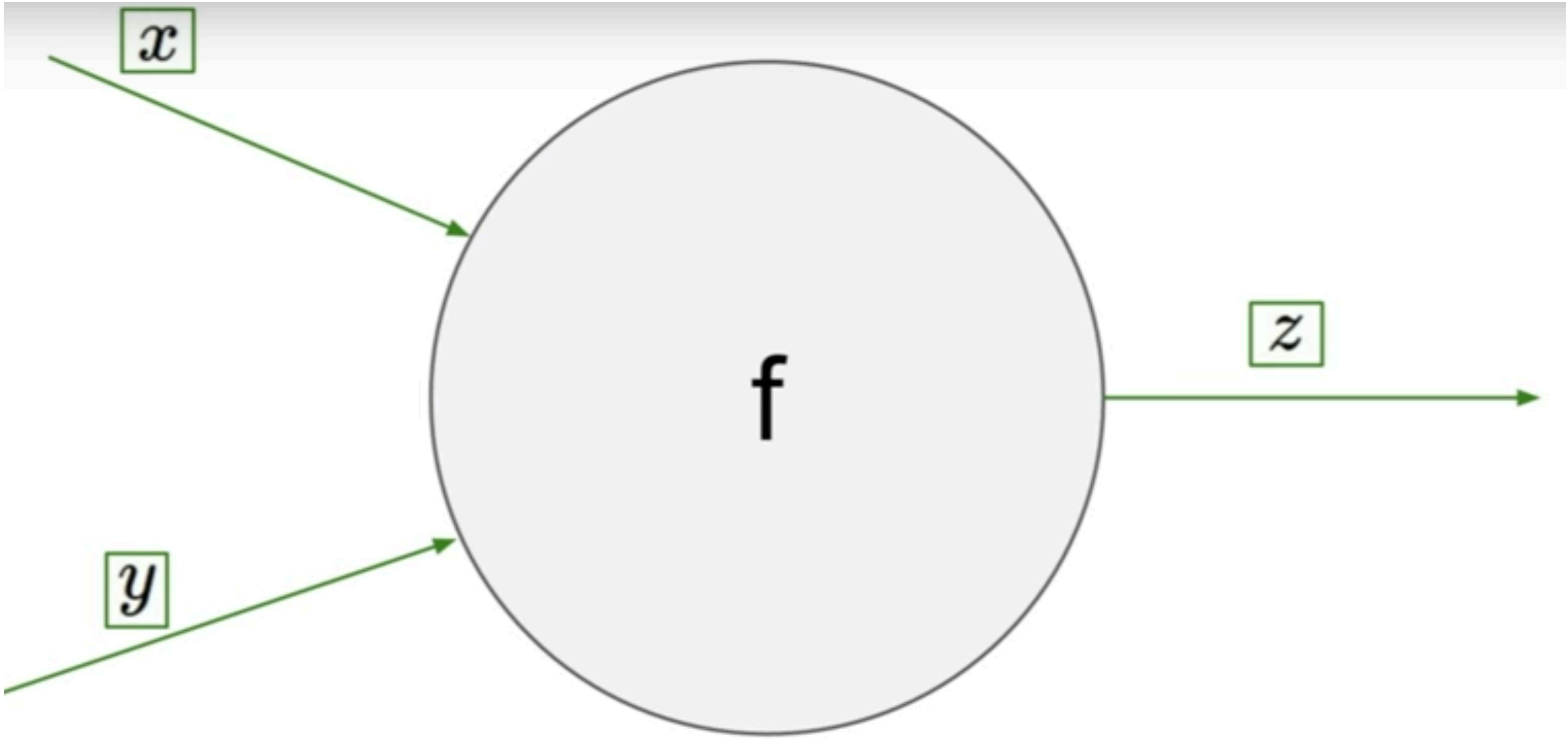


Gradient Descent Algorithm
for back propagation



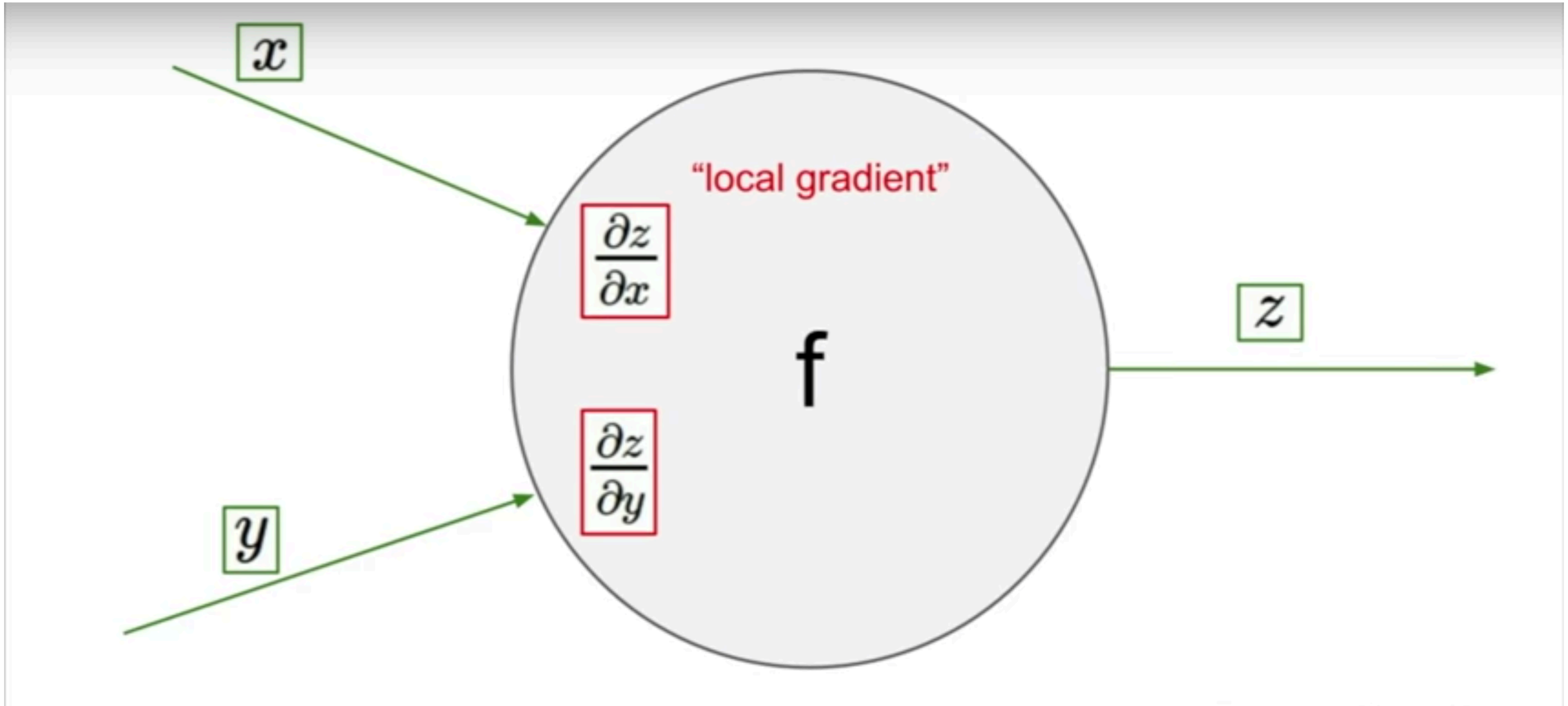
Back Propagation

Slides courtesy: Stanford Online Course



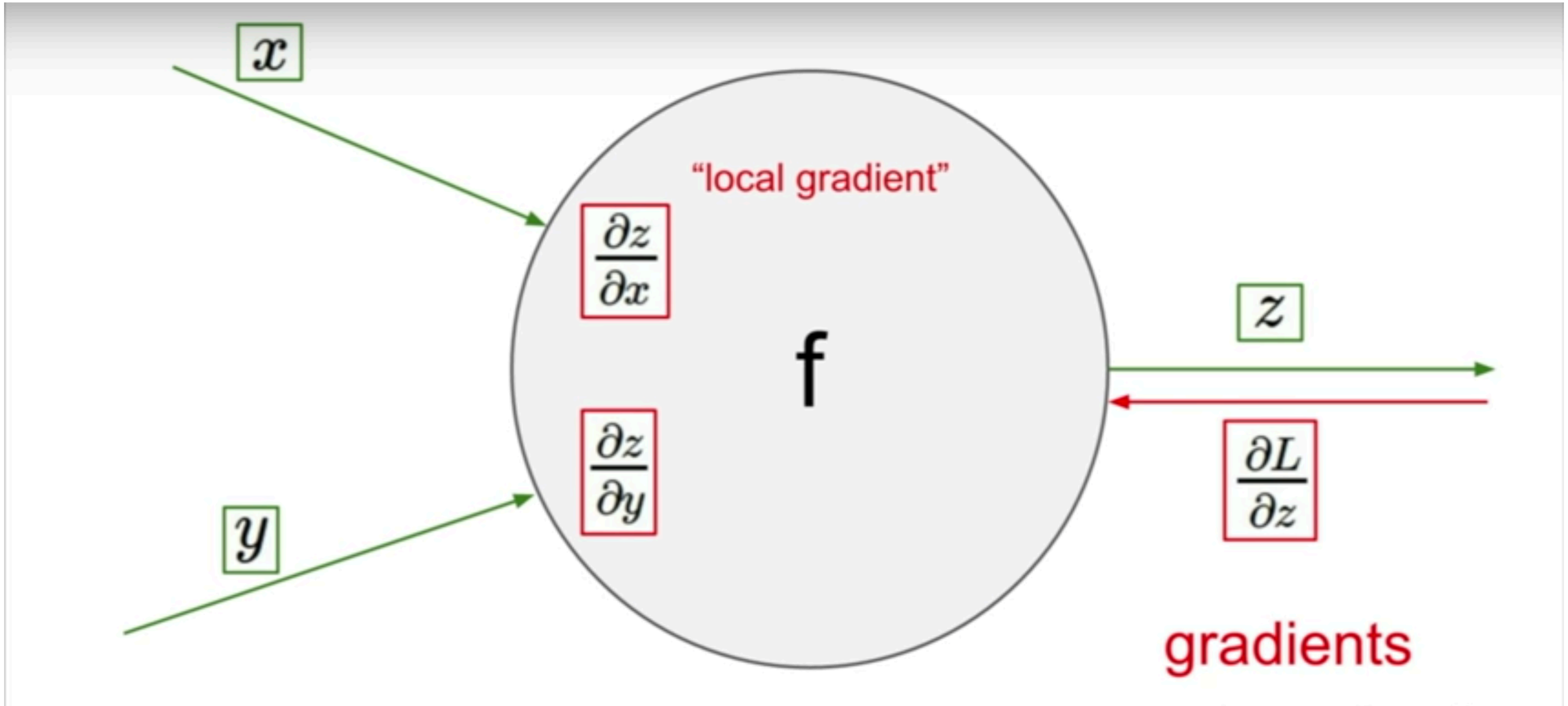
Back Propagation

Slides courtesy: Stanford Online Course



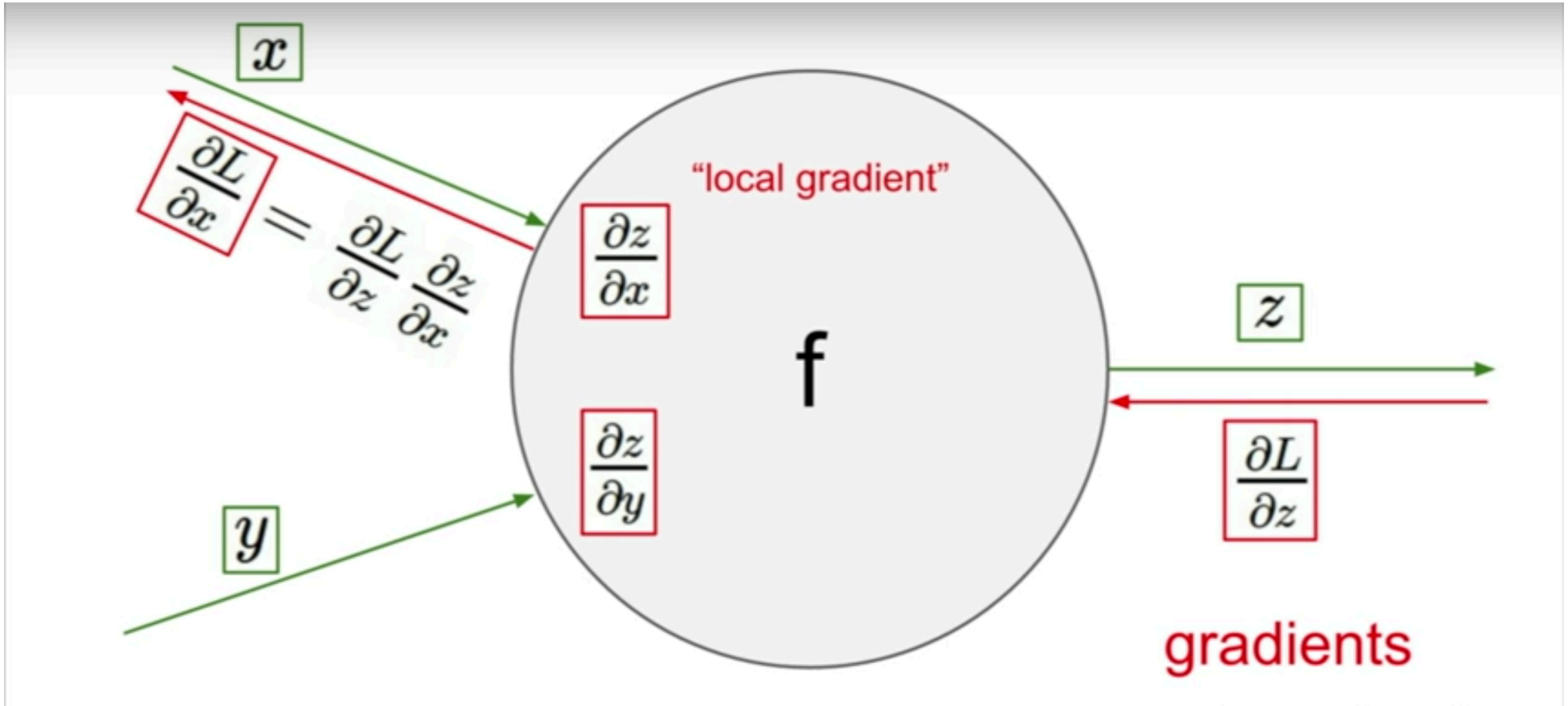
Back Propagation

Slides courtesy: Stanford Online Course



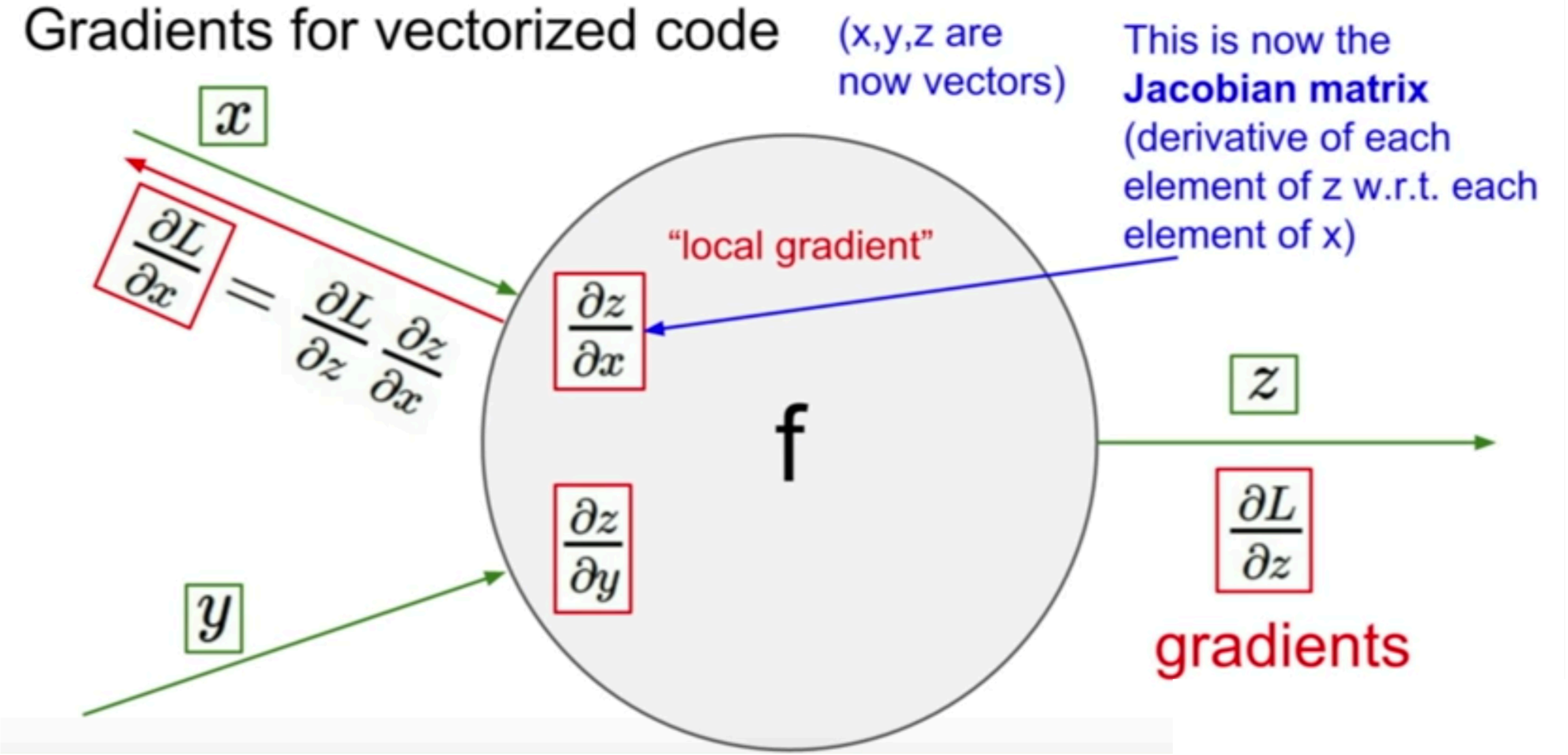
Back Propagation

Slides courtesy: Stanford Online Course



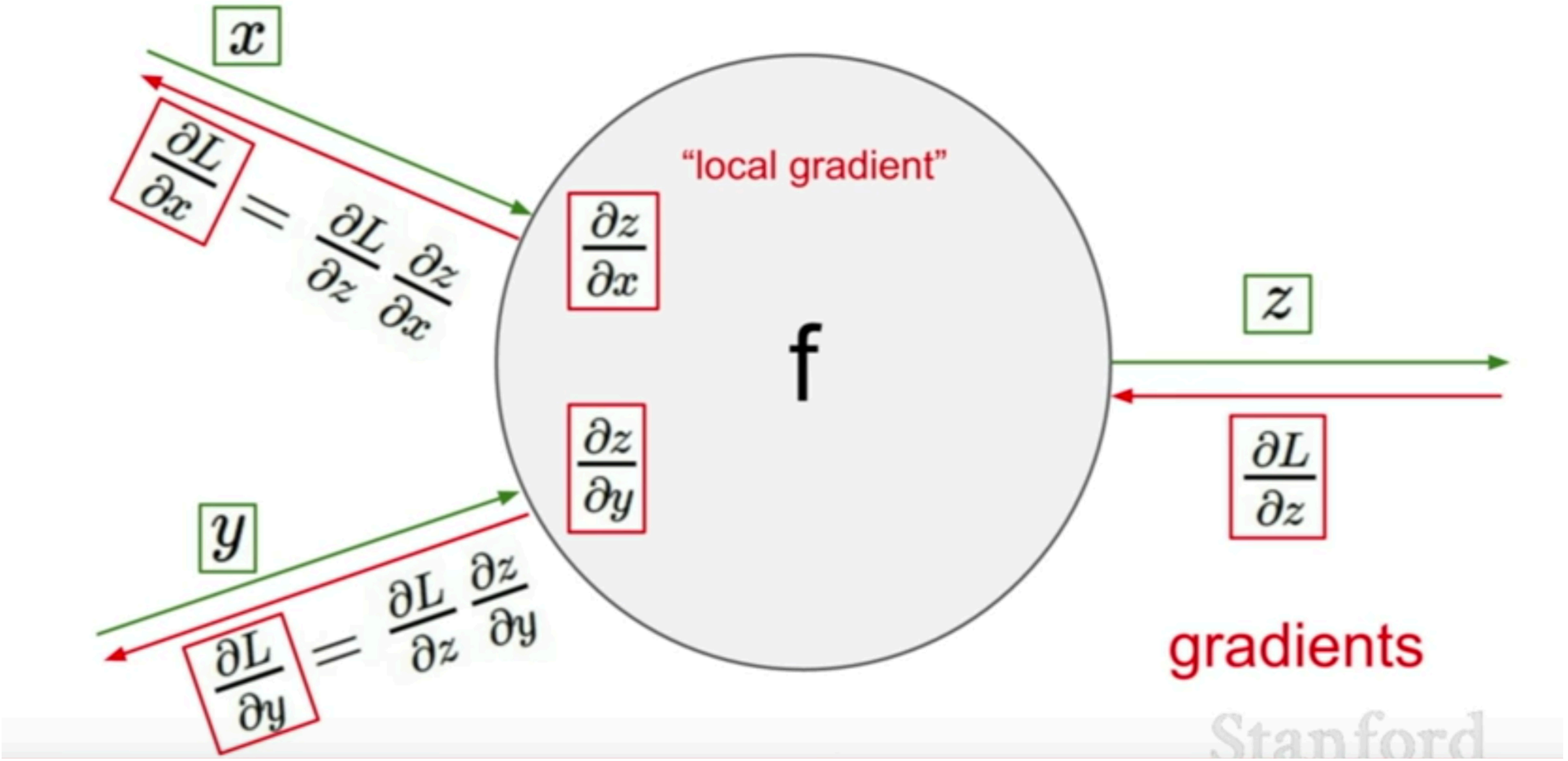
Back Propagation

Slides courtesy: Stanford Online Course



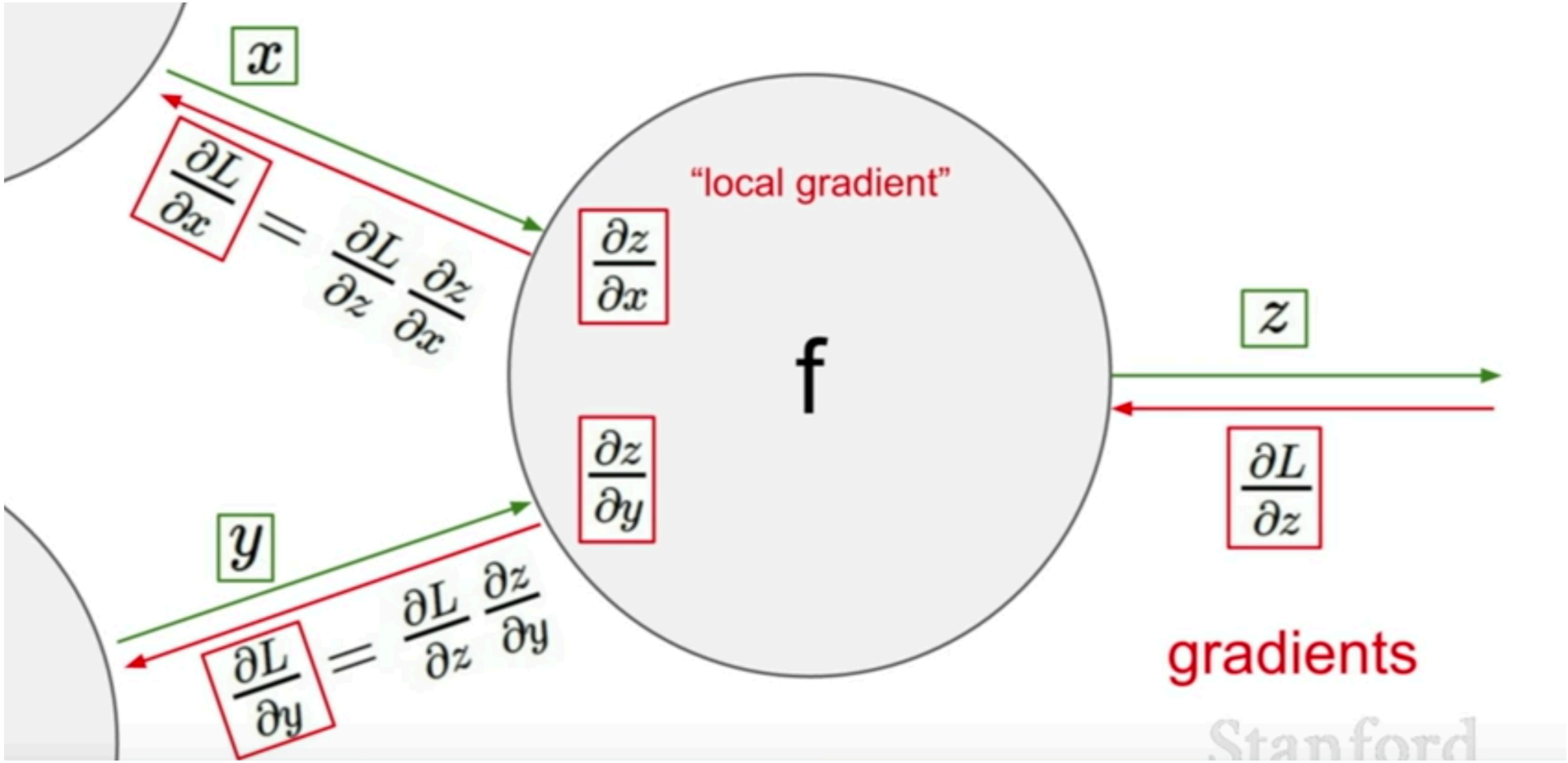
Back Propagation

Slides courtesy: <u>Stanford Online Course</u>



Back Propagation

Slides courtesy: Stanford Online Course



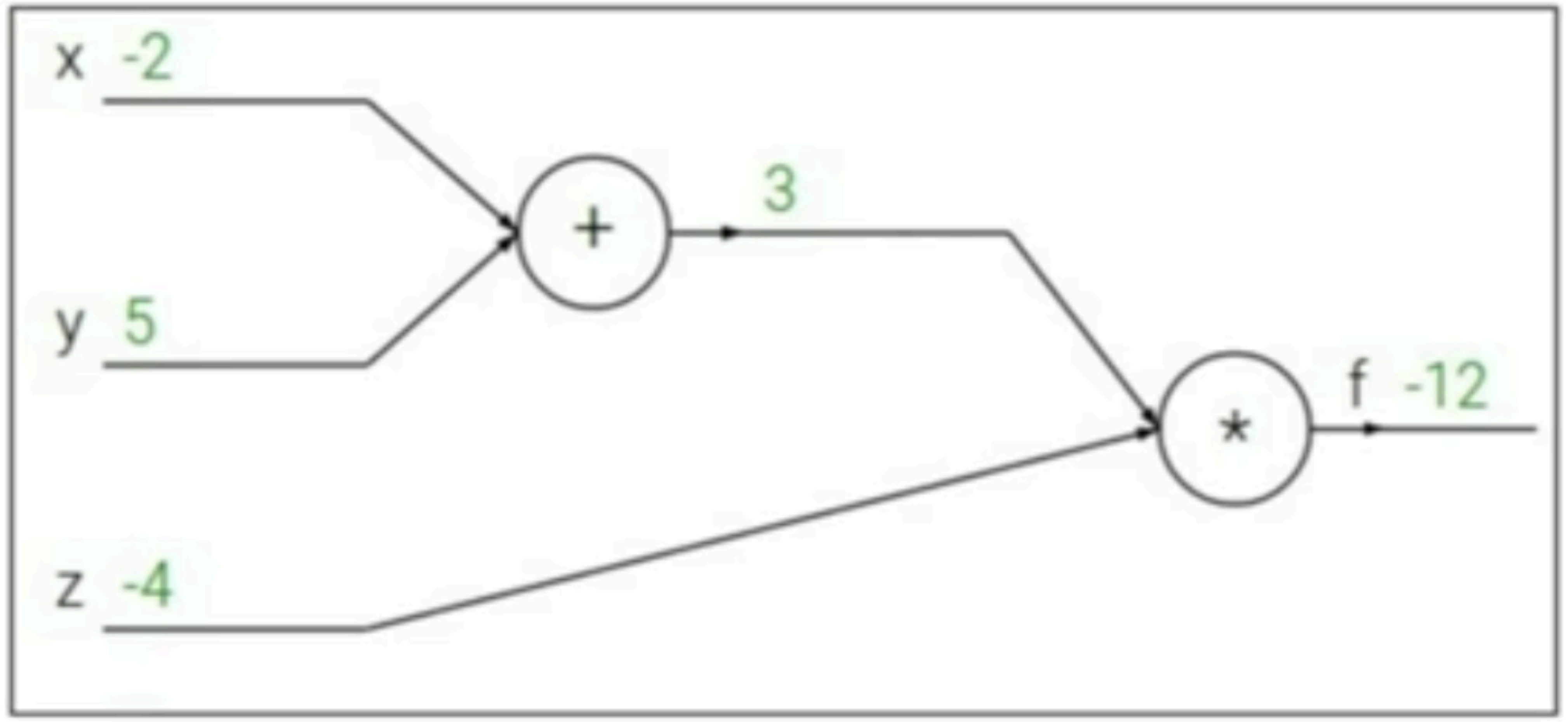
Back Propagation

Slides courtesy: Stanford Online Course

Backpropagation: a simple example

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

e.g. x = -2, y = 5, z = -4



Back Propagation

Slides courtesy: Stanford Online Course

Backpropagation: a simple example

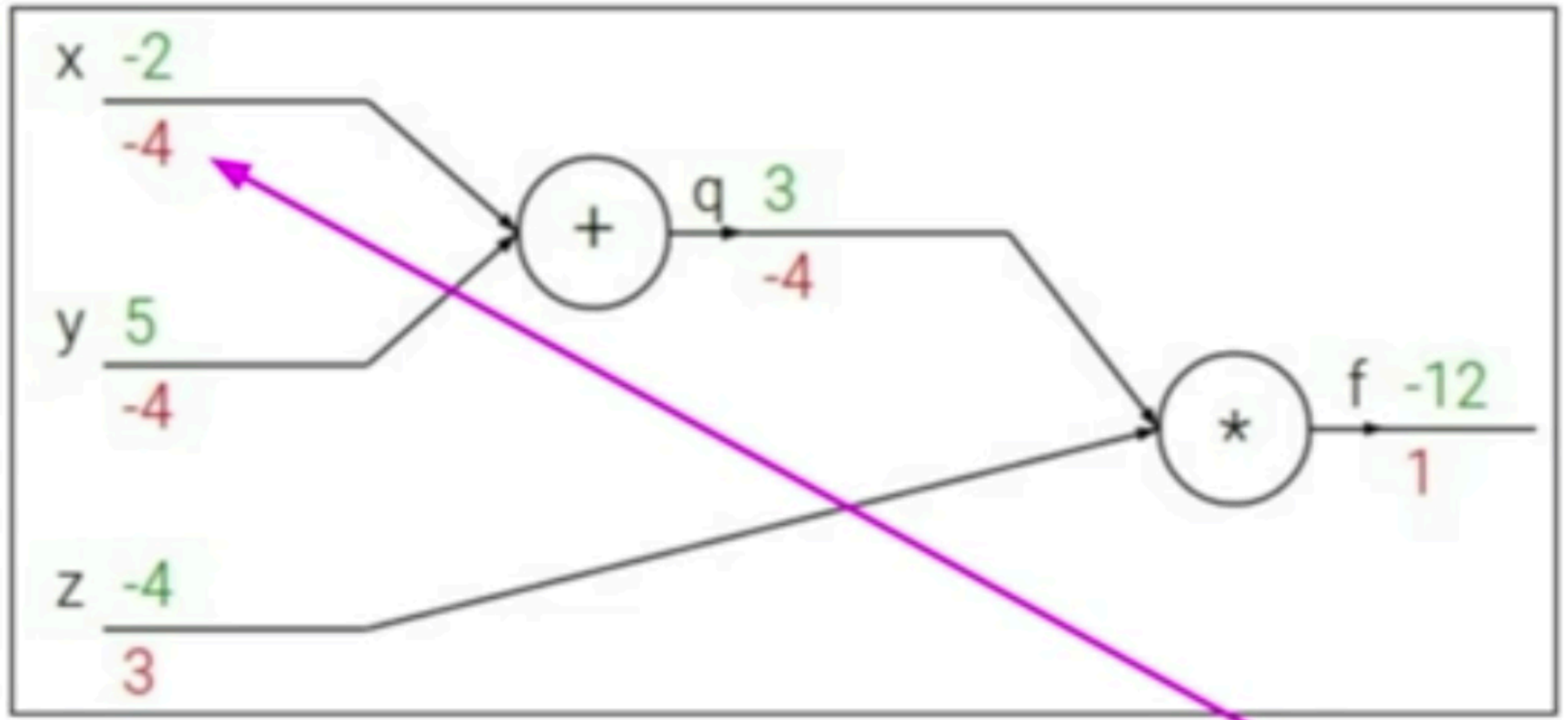
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial x}$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

UNIVERSITÄT DES SAARLANDES

Machine Learning for Filtering Monte Carlo Noise

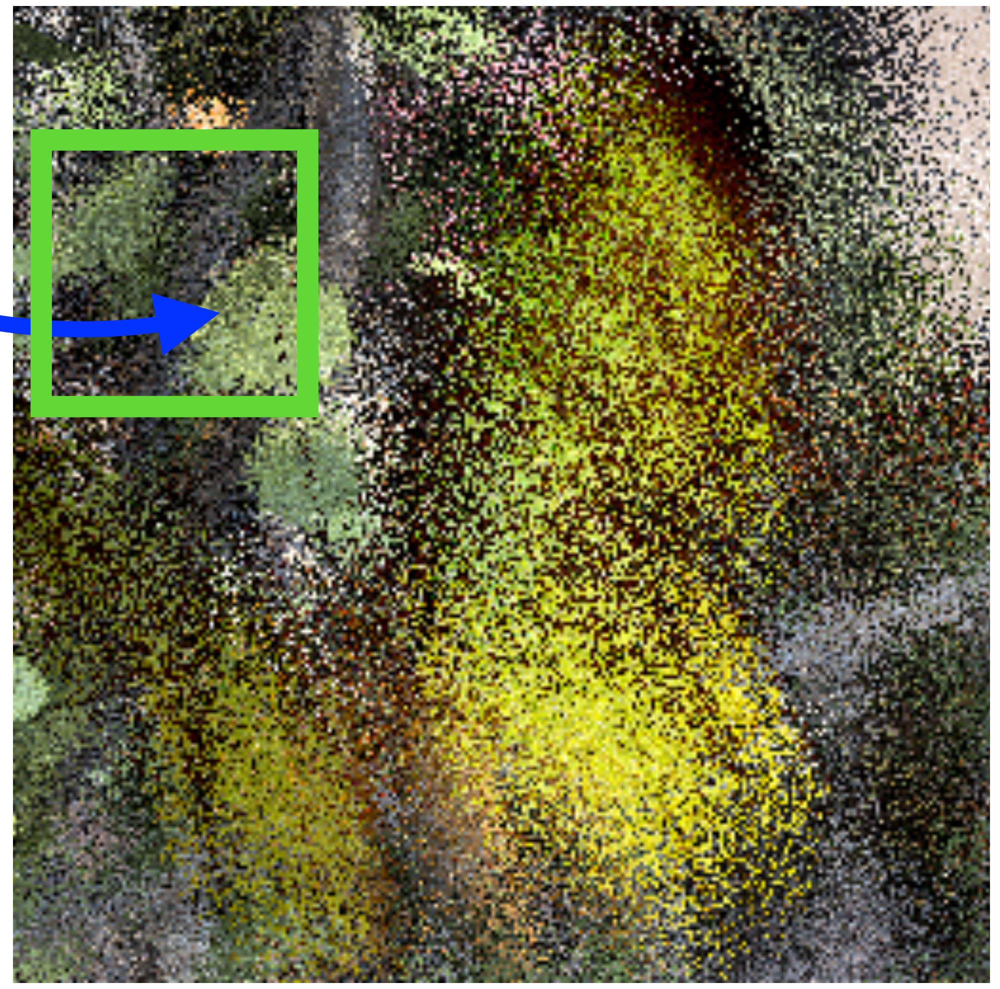
Kalantari et al. [SIGGRAPH 2015]

Reconstruction / Denoising

Filter weights

$$\hat{\mathbf{c}}_i = \frac{\sum_{j \in \mathcal{N}(i)} d_{i,j} \bar{\mathbf{c}}_j}{\sum_{j \in \mathcal{N}(i)} d_{i,j}}, \quad \hat{\mathbf{c}} = \{\hat{c}_r, \hat{c}_g, \hat{c}_b\}$$

Pixel neighborhood



Filter weights

Filter weights

$$\hat{\mathbf{c}}_i = \frac{\sum_{j \in \mathcal{N}(i)} d_{i,j} \bar{\mathbf{c}}_j}{\sum_{j \in \mathcal{N}(i)} d_{i,j}}$$

Pixel neighborhood

For cross Bilateral filters:

$$d_{i,j} = \exp\left[-\frac{\|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j\|^2}{2\alpha_i^2}\right] \times \exp\left[-\frac{D(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_j)}{2\beta_i^2}\right] \times \prod_{k=1}^{K} \exp\left[-\frac{D_k(\bar{\mathbf{f}}_{i,k}, \bar{\mathbf{f}}_{j,k})}{2\gamma_{k,i}^2}\right],$$

Filter weights

Filter weights

$$\hat{\mathbf{c}}_i = \frac{\sum_{j \in \mathcal{N}(i)} d_{i,j} \bar{\mathbf{c}}_j}{\sum_{j \in \mathcal{N}(i)} d_{i,j}}$$

Pixel neighborhood

For cross Bilateral filters:

$$d_{i,j} = \exp\left[-\frac{\|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j\|^2}{2\alpha_i^2}\right] \times \exp\left[-\frac{D(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_j)}{2\beta_i^2}\right] \times \prod_{k=1}^{K} \exp\left[-\frac{D_k(\bar{\mathbf{f}}_{i,k}, \bar{\mathbf{f}}_{j,k})}{2\gamma_{k,i}^2}\right],$$

Filter weights

Filter weights

$$\hat{\mathbf{c}}_i = \frac{\sum_{j \in \mathcal{N}(i)} d_{i,j} \bar{\mathbf{c}}_j}{\sum_{j \in \mathcal{N}(i)} d_{i,j}}$$

Pixel neighborhood

For cross Bilateral filters:

$$d_{i,j} = \exp\left[-\frac{\|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j\|^2}{2\alpha_i^2}\right] \times \exp\left[-\frac{D(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_j)}{2\beta_i^2}\right] \times \prod_{k=1}^{K} \exp\left[-\frac{D_k(\bar{\mathbf{f}}_{i,k}, \bar{\mathbf{f}}_{j,k})}{2\gamma_{k,i}^2}\right],$$

Sen and Darabi [2012]

Filter weights

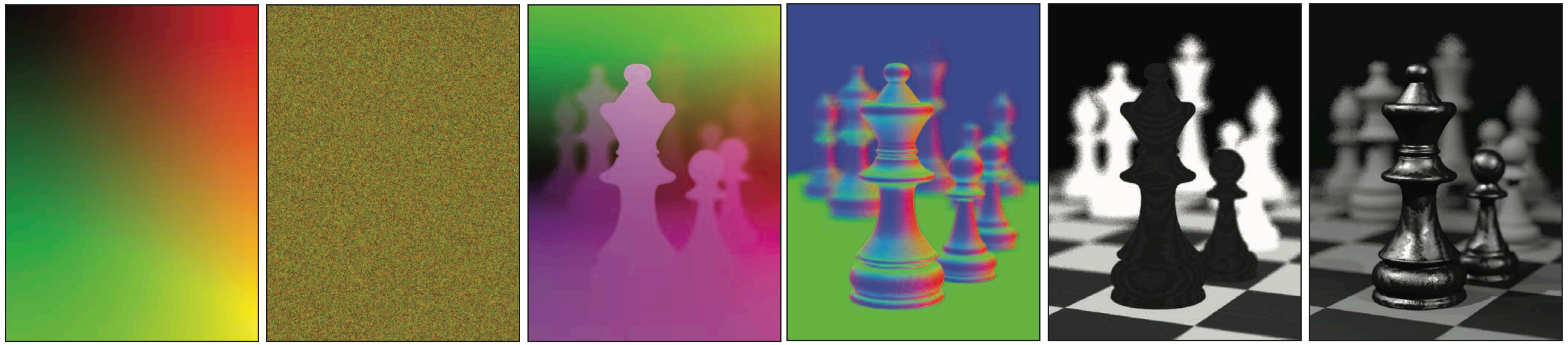
For cross Bilateral filters:

$$d_{i,j} = \exp\left[-\frac{\|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j\|^2}{2\alpha_i^2}\right] \times \exp\left[-\frac{D(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_j)}{2\beta_i^2}\right] \times \prod_{k=1}^{K} \exp\left[-\frac{D_k(\bar{\mathbf{f}}_{i,k}, \bar{\mathbf{f}}_{j,k})}{2\gamma_{k,i}^2}\right],$$

Pixel screen coordinates

Mean sample color value

Scene features



(a) Screen position (b) Random parameters (c) World space coords. (d) Surface normals (e) Texture value (f) Sample color

Filter weights

For cross Bilateral filters:

$$d_{i,j} = \exp\left[-\frac{\|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j\|^2}{2\alpha_i^2}\right] \times \exp\left[-\frac{D(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_j)}{2\beta_i^2}\right] \times \prod_{k=1}^{K} \exp\left[-\frac{D_k(\bar{\mathbf{f}}_{i,k}, \bar{\mathbf{f}}_{j,k})}{2\gamma_{k,i}^2}\right],$$

Pixel screen coordinates

Mean sample color value

Scene features

What are the **optimal** parameters ?

Neural Network Approach

- Feed-forward Neural network
- Best part: We can learn weights in a training phase
- Back propagation: Important for training weights
- For Back propagation, the Loss function should be differentiable and
- all the intermediate functionals should be differentiable.

One Hidden-layer model

Relative Mean Square Error:

$$E_i = \frac{n}{2} \sum_{q \in \{r,g,b\}} \frac{\left(\hat{c}_{i,q} - c_{i,q}\right)^2}{c_{i,q}^2 + \varepsilon}$$

One Hidden-layer model

Relative Mean Square Error:

$$\frac{\partial E_i}{\partial w^l_{t,s}} = \sum_{m=1}^{M} \left[\sum_{q \in \{r,g,b\}} \left[\frac{\partial E_{i,q}}{\partial \hat{c}_{i,q}} \frac{\partial \hat{c}_{i,q}}{\partial \theta_{m,i}} \right] \frac{\partial \theta_{m,i}}{\partial w^l_{t,s}} \right]$$

$$E_i = \frac{n}{2} \sum_{q \in \{r,g,b\}} \frac{\left(\hat{c}_{i,q} - c_{i,q}\right)^2}{c^2_{i,q} + \varepsilon}$$

$$\frac{\partial E_i}{\partial \hat{c}_{i,q}} = ???$$

One Hidden-layer model

Relative Mean Square Error:

$$\frac{\partial E_i}{\partial w_{t,s}^l} = \sum_{m=1}^{M} \left[\sum_{q \in \{r,g,b\}} \left[\frac{\partial E_{i,q}}{\partial \hat{c}_{i,q}} \frac{\partial \hat{c}_{i,q}}{\partial \theta_{m,i}} \right] \frac{\partial \theta_{m,i}}{\partial w_{t,s}^l} \right]$$

$$E_i = \frac{n}{2} \sum_{q \in \{r,g,b\}} \frac{\left(\hat{c}_{i,q} - c_{i,q}\right)^2}{c_{i,q}^2 + \varepsilon}$$

$$\frac{\partial E_i}{\partial \hat{c}_{i,q}} = n \frac{\hat{c}_{i,q} - c_{i,q}}{c_{i,q}^2 + \epsilon}$$

Results

Our result with a cross-bilateral filter (4 spp)

Our result with a non-local means filter (4 spp)