Deep Learning Mathematics

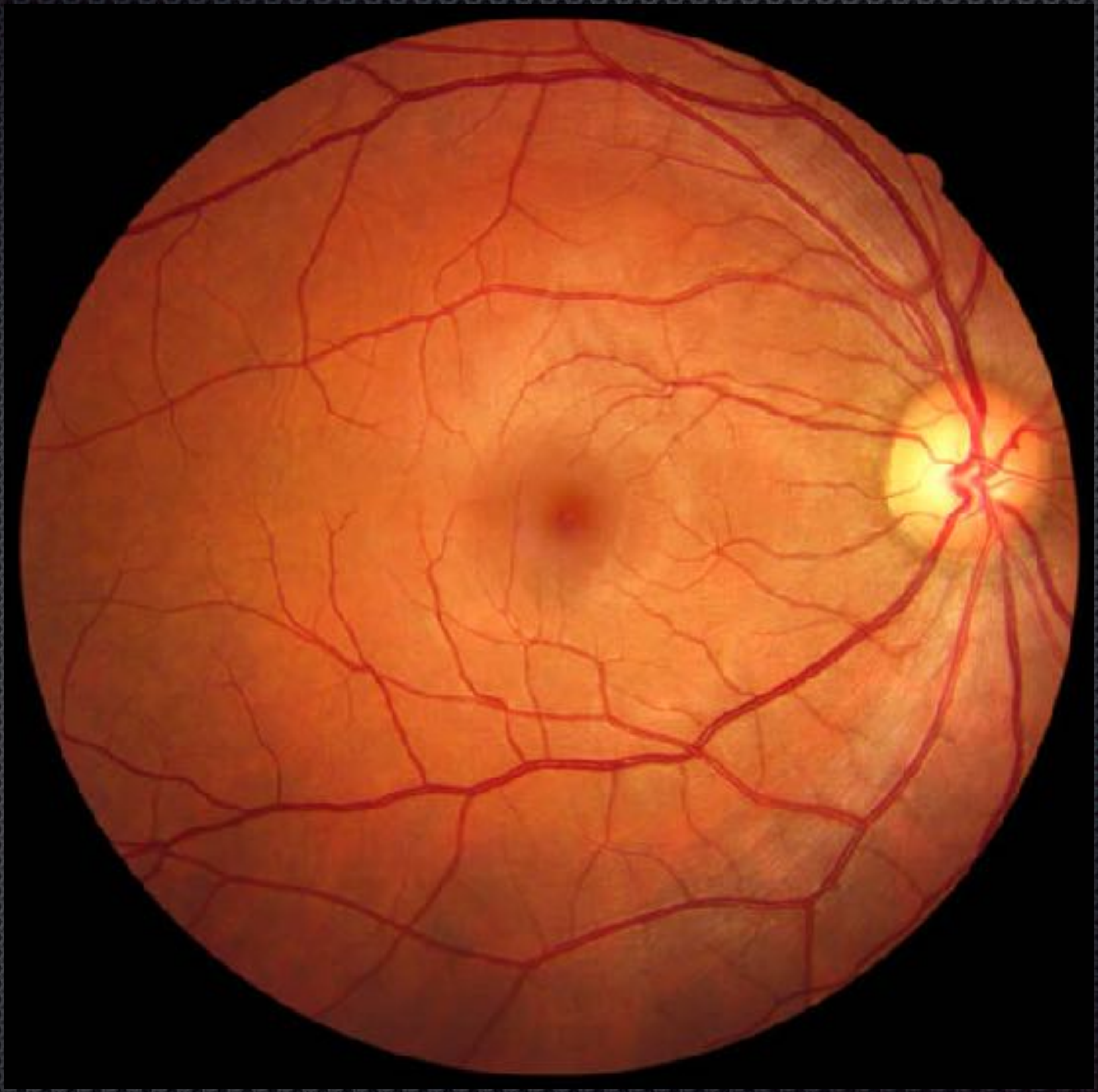
Data Science Nigeria AI Bootcamp
Lagos, Nigeria

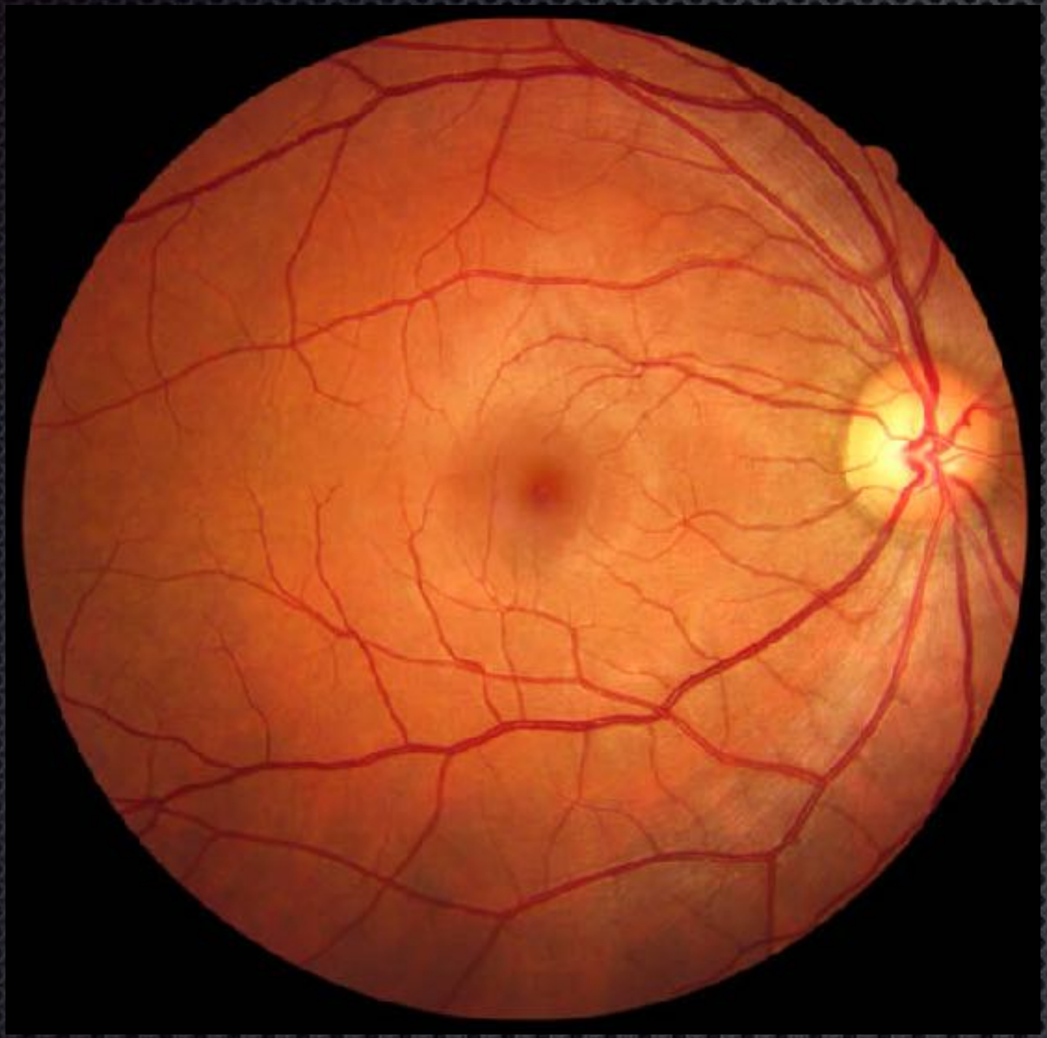
Stephen G. Odaibo
M.D., M.S.(Math), M.S.(Comp. Sci.)

CEO, Founder, & Chief Software Architect
RETINA-AI Health, Inc.
Houston Texas

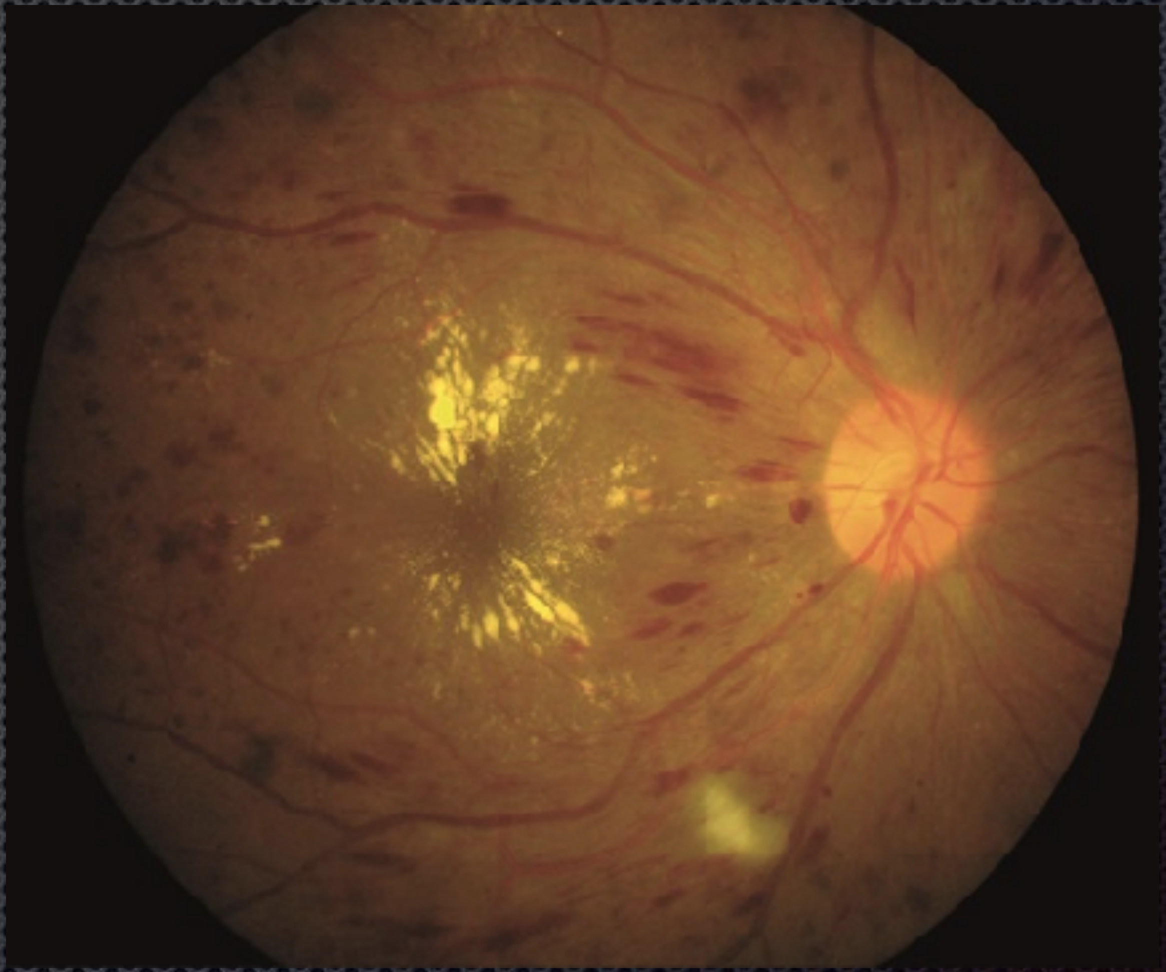
Faculty Member
MD Anderson Cancer Center

RETINA-AI





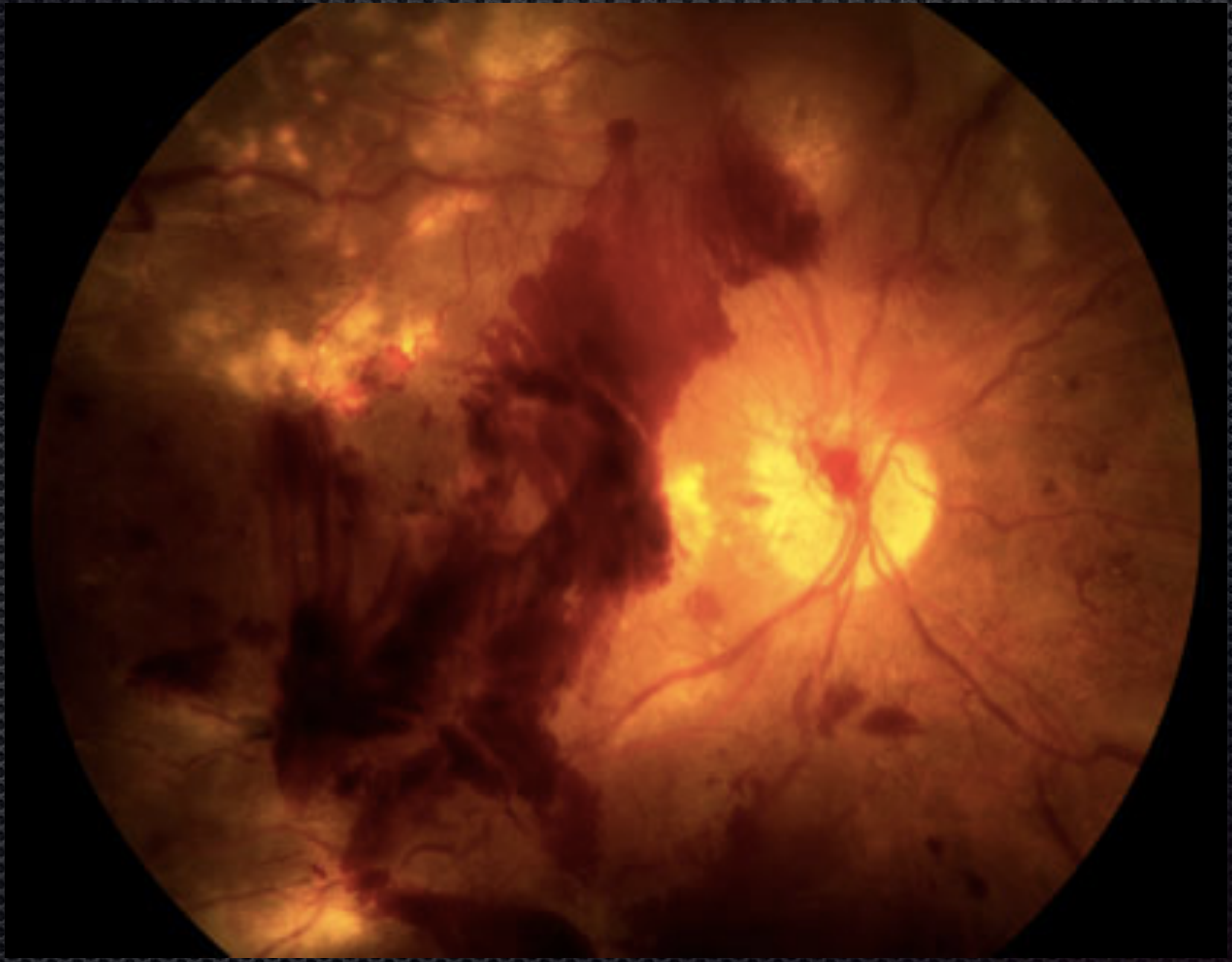
Normal Retina



Diabetic Retinopathy

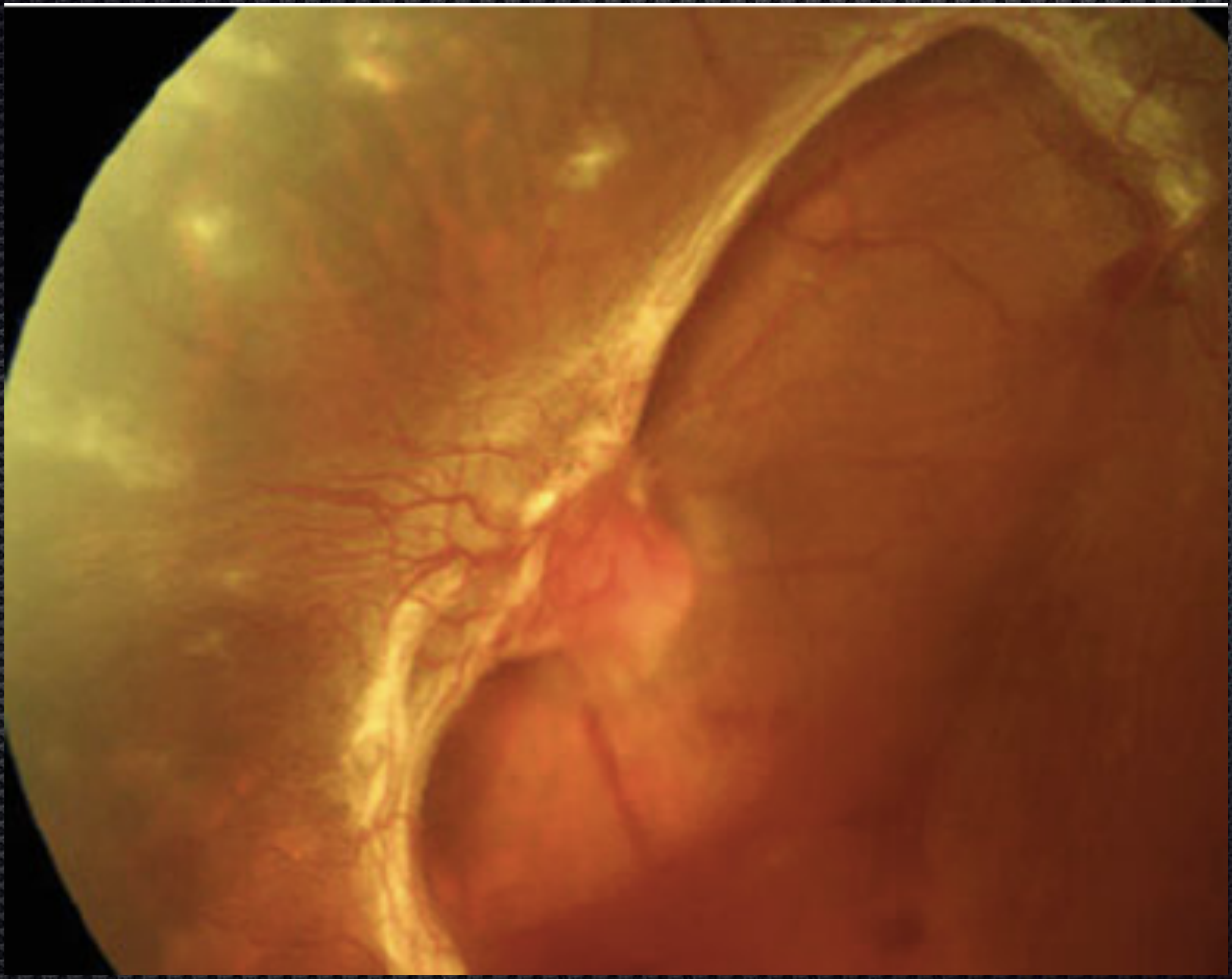
RETINA-AI

Diabetic Vitreous Hemorrhage



RETINA-AI

Diabetic Retinal Detachment

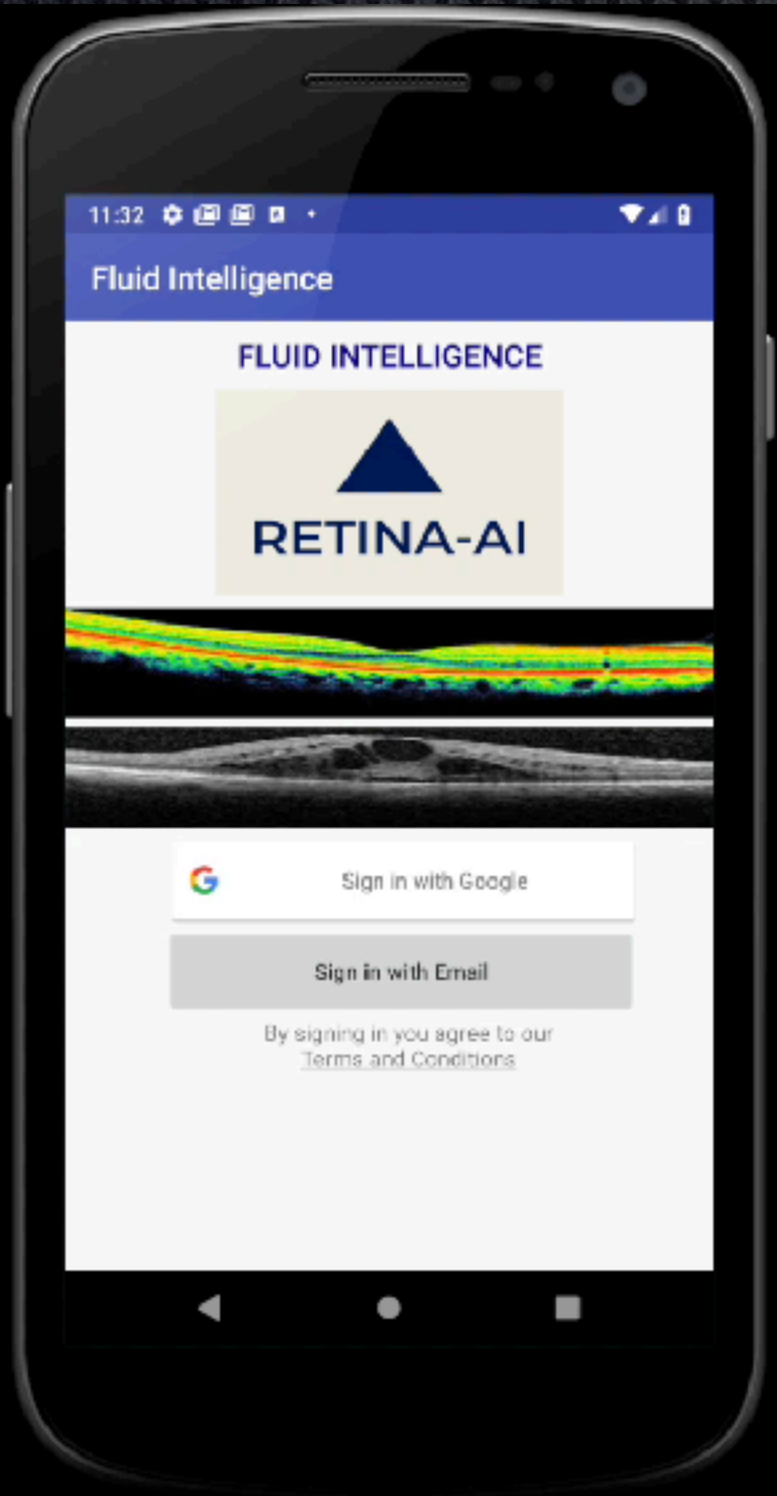


RETINA-AI

Numbers

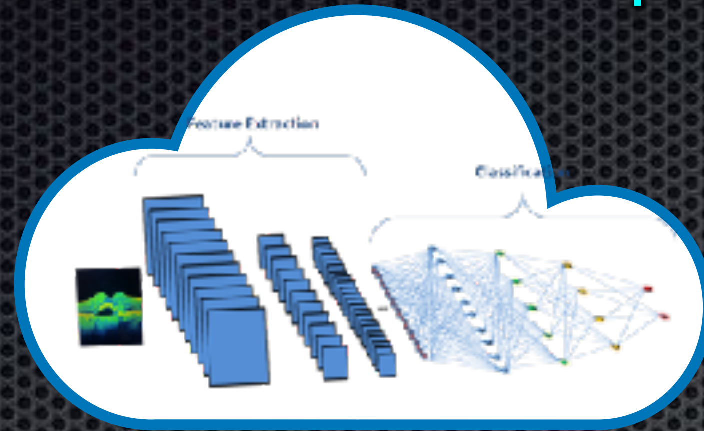
- 350-500 Ophthalmologists (<600)
- 200 million people
- 5-10% diabetes prevalence:
- 20,000-40,000 diabetics/ophthalmologist

RETINA-AI


11:32
Fluid Intelligence
FLUID INTELLIGENCE
RETINA-AI
Sign in with Google
Sign in with Email
By signing in you agree to our
Terms and Conditions

RETINA-AI

Cloud-Hosted AI Interpretation



RETINA-AI interpretation & Cloud storage

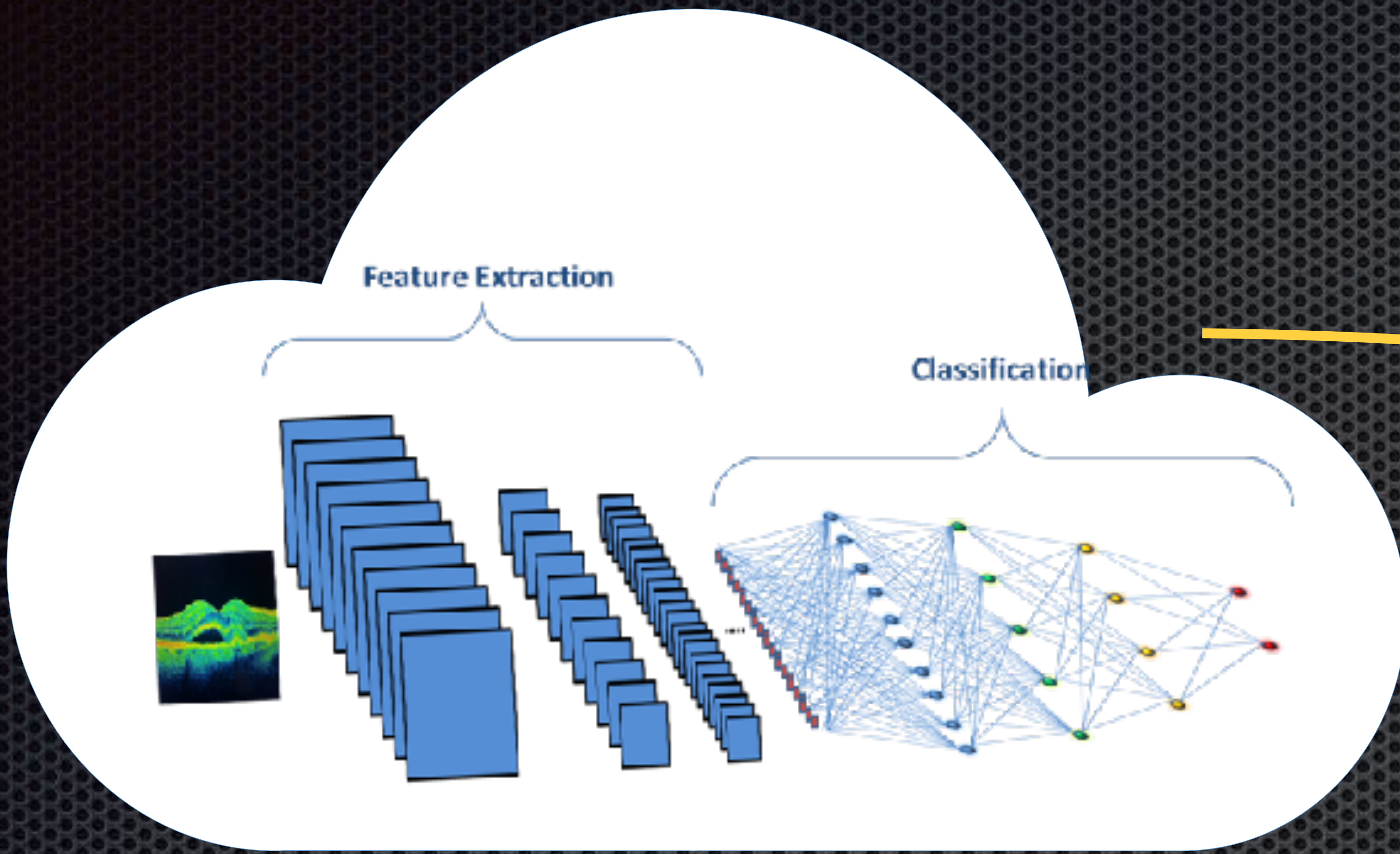
- ☑ Fast
- ☐ No Internet required
- ☑ Access to continuous Learning

Fundus Camera

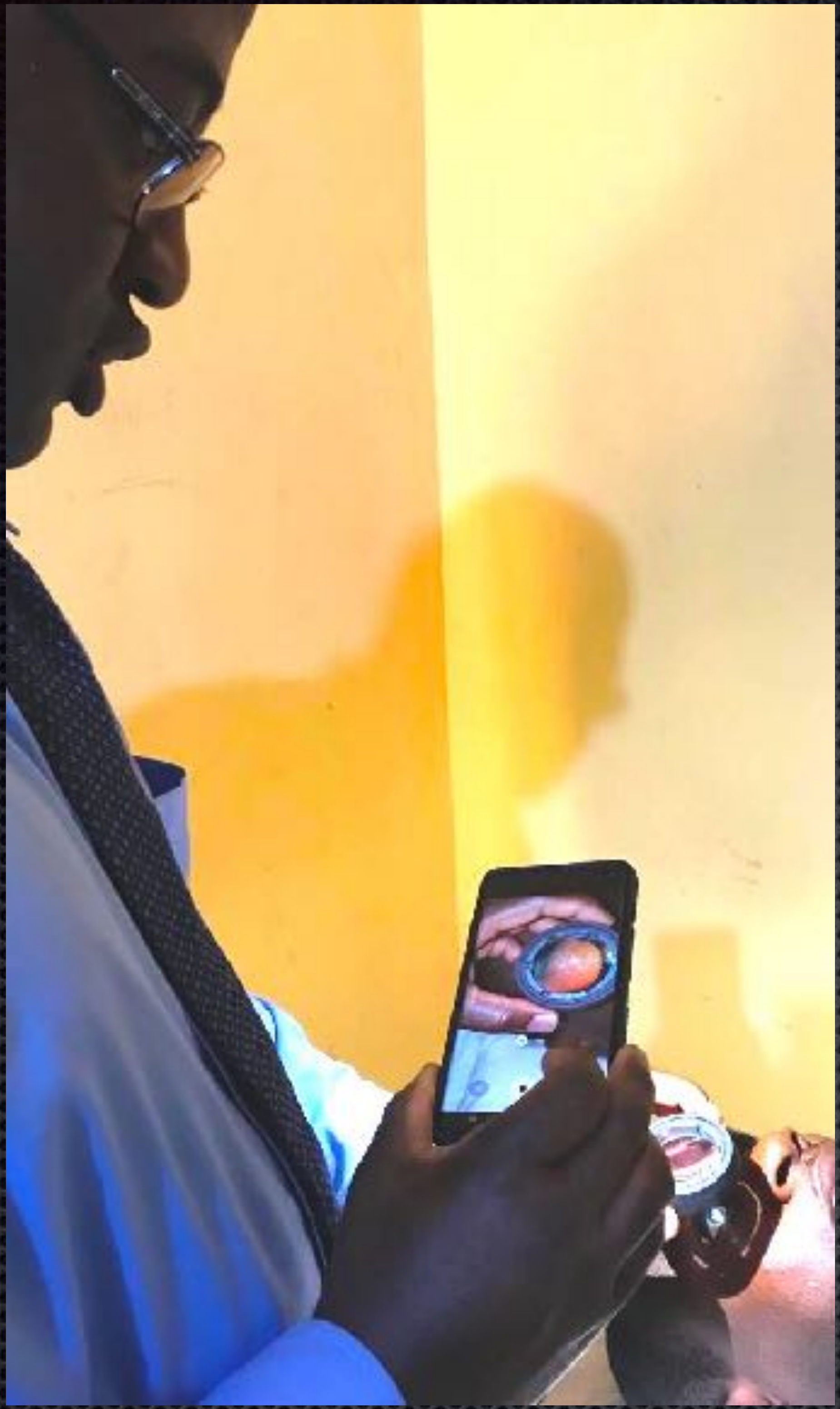
Immediate ✔

EMR

“Inference”

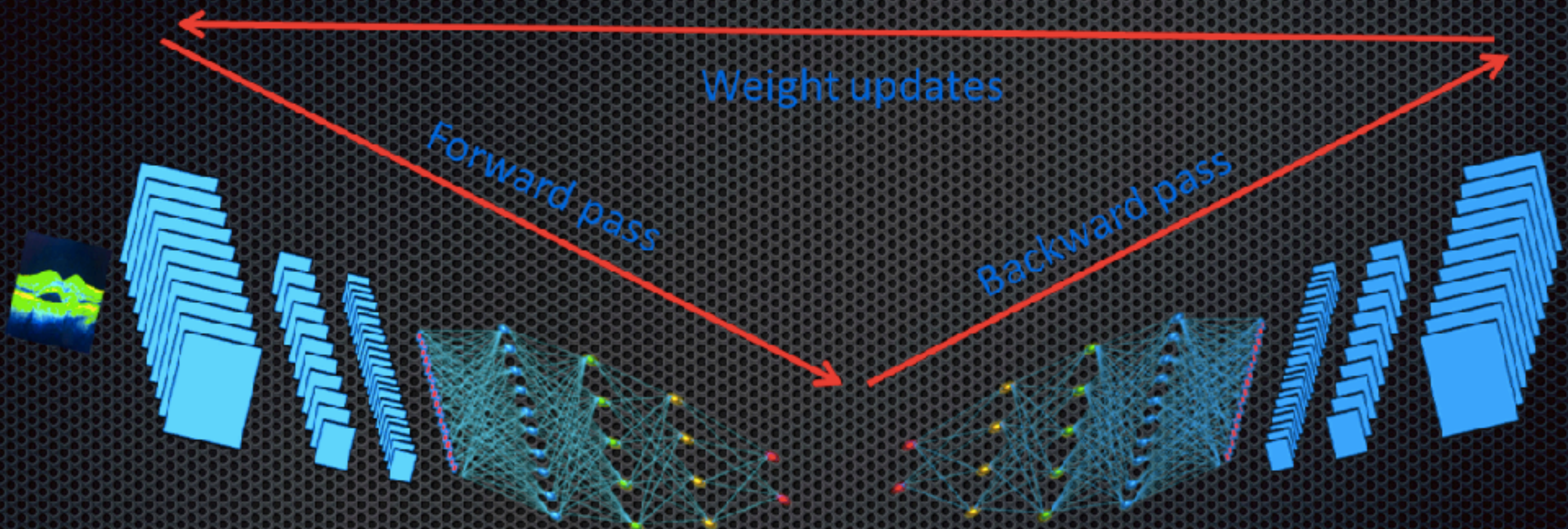


RETINA-AI

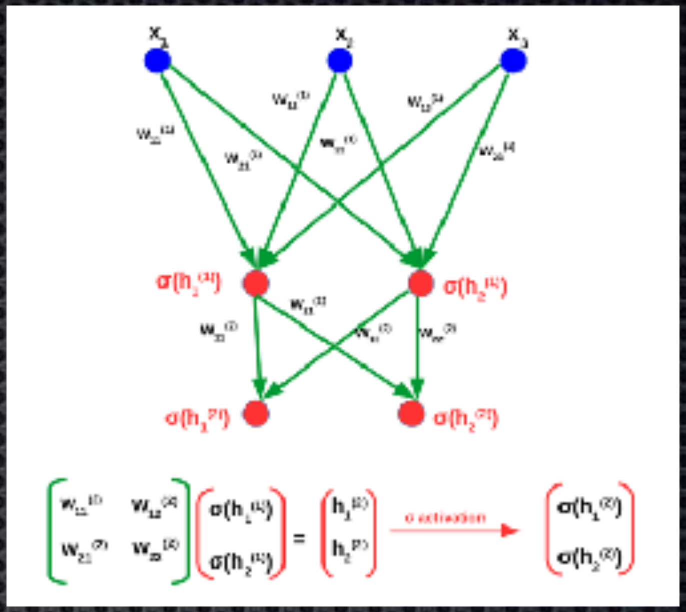


RETINA-AI

Convolutional Neural Networks

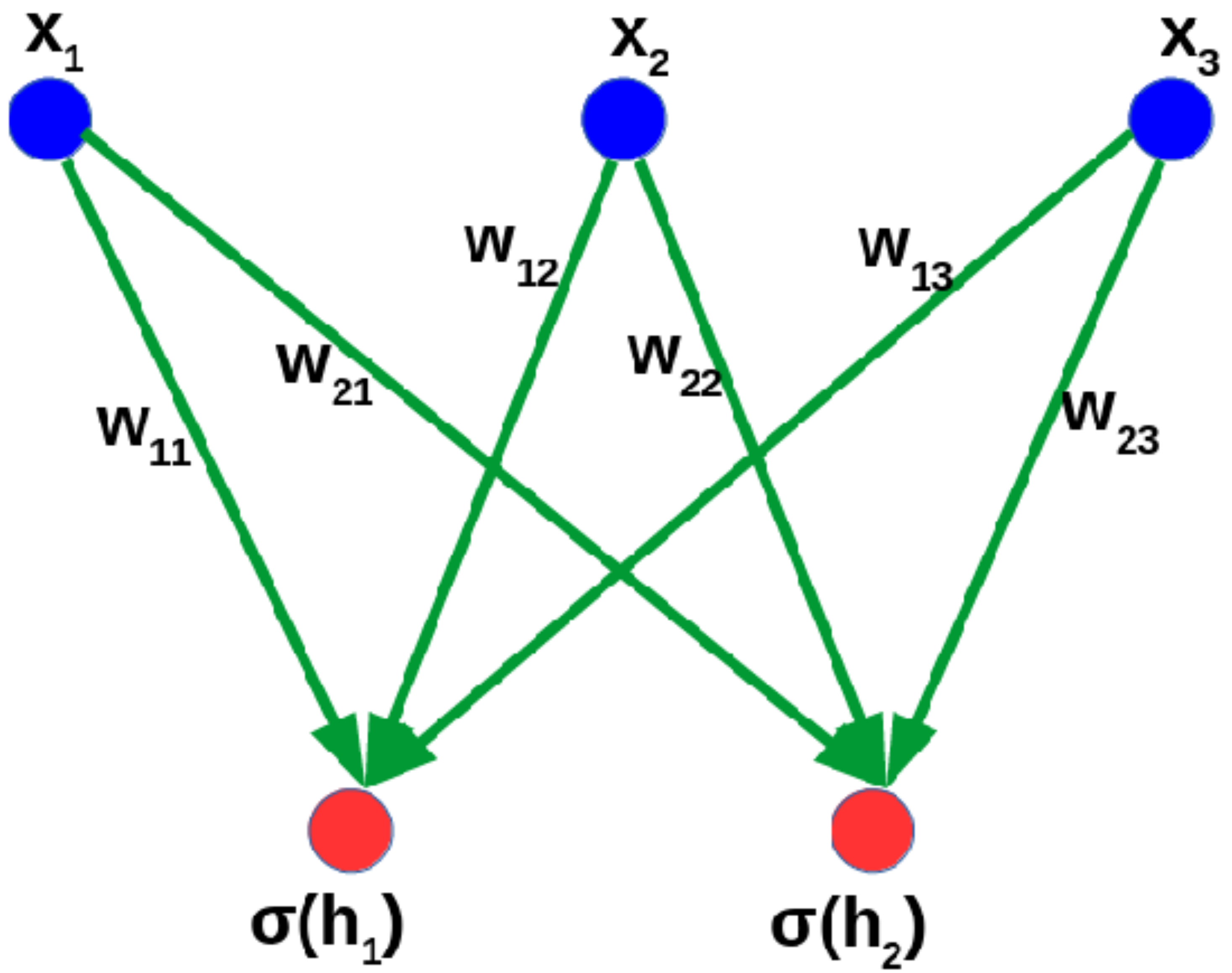


- Neural Network Anatomy (Structure):
  - Input data
  - Nodes
  - Weights
  - Activations
  - Loss

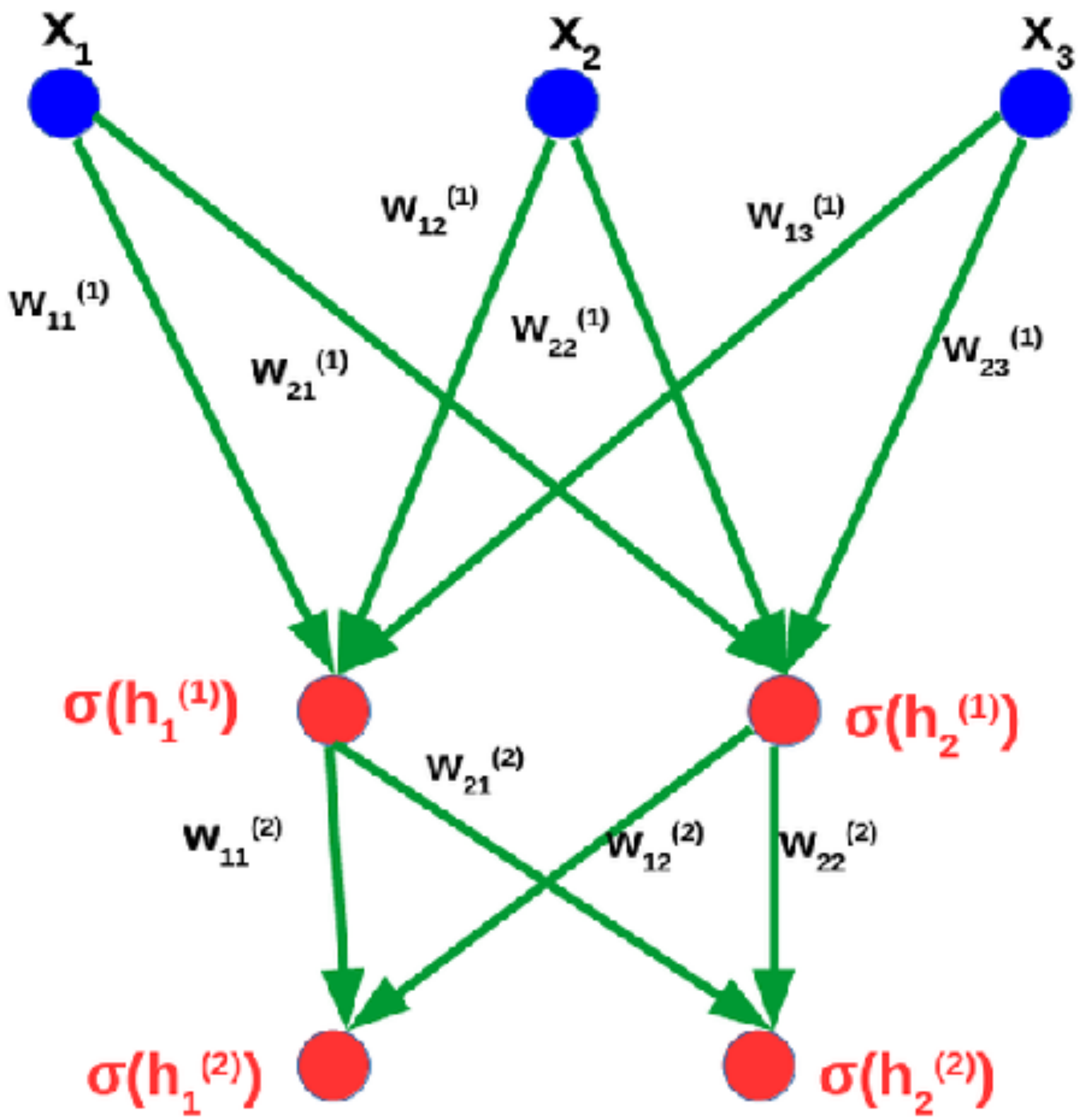


- Neural Network Physiology (Function):
  - Stochastic gradient descent
    - Backpropagation

RETINA-AI

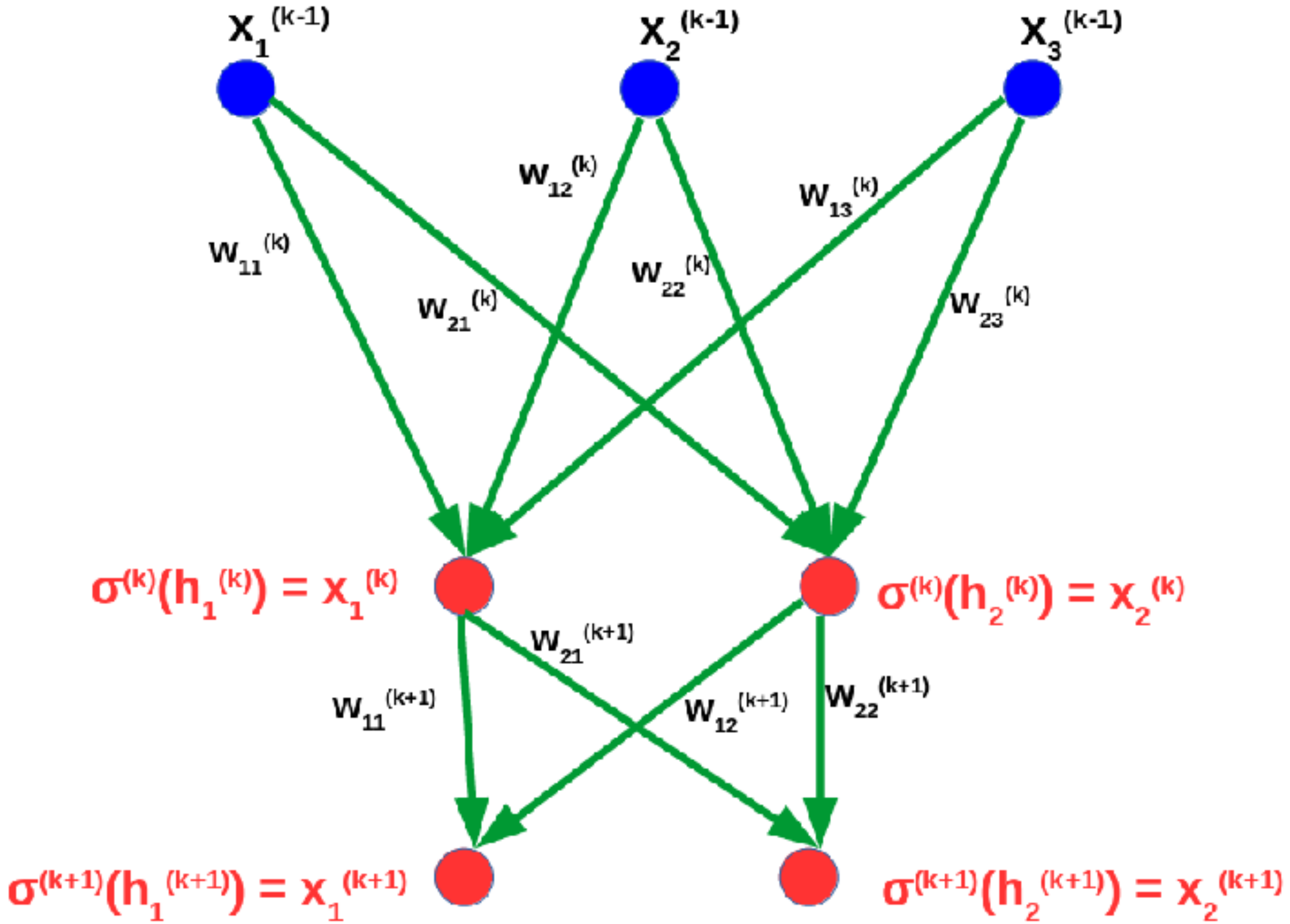


$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \xrightarrow{\sigma \text{ activation}} \begin{bmatrix} \sigma(h_1) \\ \sigma(h_2) \end{bmatrix}$$

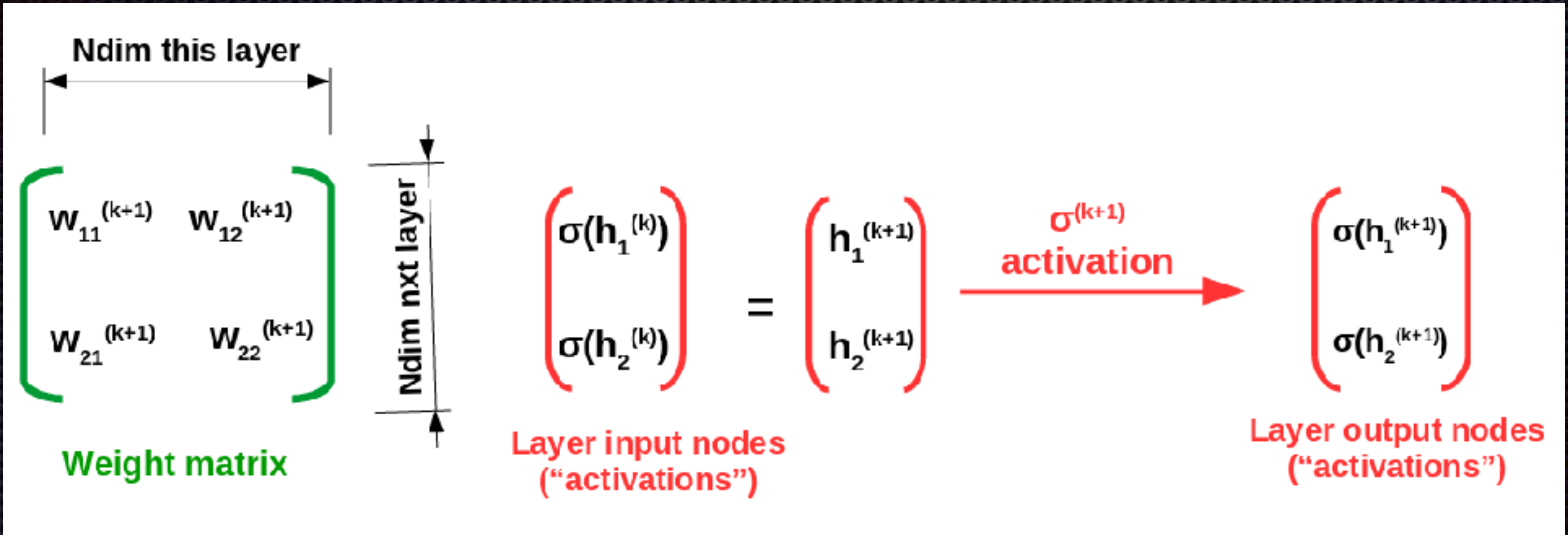


$$\begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix} \begin{pmatrix} \sigma(h_1^{(1)}) \\ \sigma(h_2^{(1)}) \end{pmatrix} = \begin{pmatrix} h_1^{(2)} \\ h_2^{(2)} \end{pmatrix} \xrightarrow{\sigma \text{ activation}} \begin{pmatrix} \sigma(h_1^{(2)}) \\ \sigma(h_2^{(2)}) \end{pmatrix}$$

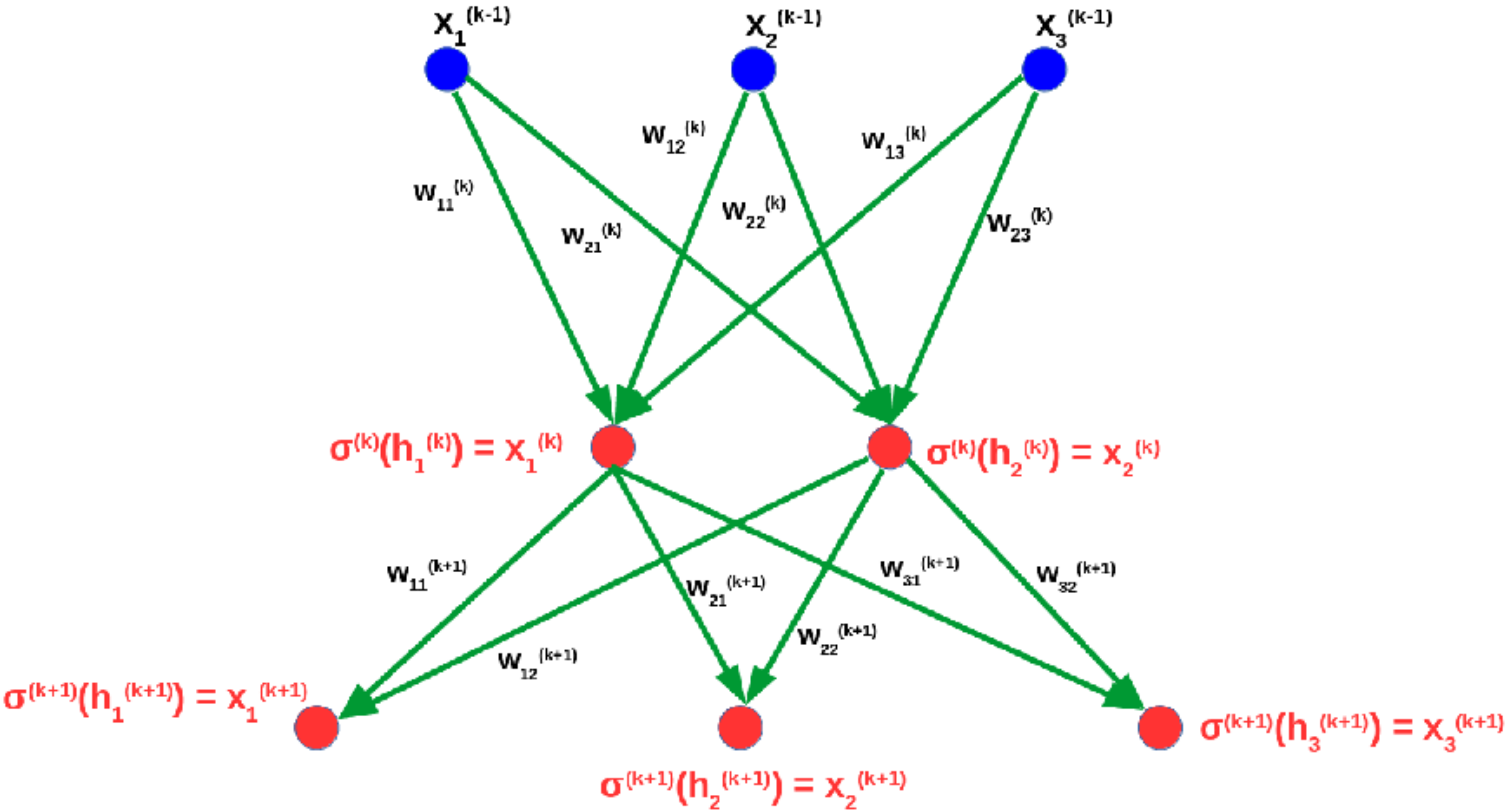
RETINA-AI



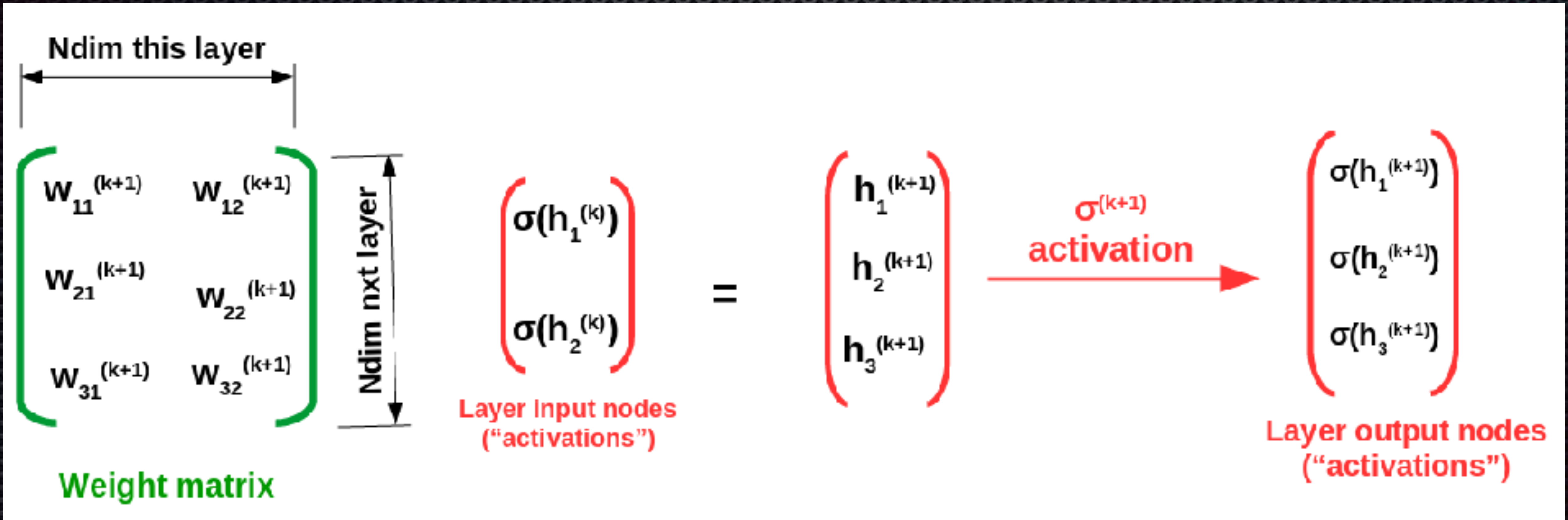
$$\begin{pmatrix} w_{11}^{(k+1)} & w_{12}^{(k+1)} \\ w_{21}^{(k+1)} & w_{22}^{(k+1)} \end{pmatrix} \begin{pmatrix} \sigma(h_1^{(k)}) \\ \sigma(h_2^{(k)}) \end{pmatrix} = \begin{pmatrix} h_1^{(k+1)} \\ h_2^{(k+1)} \end{pmatrix} \xrightarrow[\text{activation}]{\sigma^{(k+1)}} \begin{pmatrix} \sigma(h_1^{(k+1)}) \\ \sigma(h_2^{(k+1)}) \end{pmatrix}$$


Ndim this layer
Ndim nxt layer
$W_{11}^{(k+1)}$ $W_{12}^{(k+1)}$
$W_{21}^{(k+1)}$ $W_{22}^{(k+1)}$
Weight matrix
$\sigma(h_1^{(k)})$
$\sigma(h_2^{(k)})$
Layer input nodes
("activations")
=
$h_1^{(k+1)}$
$h_2^{(k+1)}$
$\sigma^{(k+1)}$
activation
$\sigma(h_1^{(k+1)})$
$\sigma(h_2^{(k+1)})$
Layer output nodes
("activations")

RETINA-AI



$$\begin{pmatrix} w_{11}^{(k+1)} & w_{12}^{(k+1)} \\ w_{21}^{(k+1)} & w_{22}^{(k+1)} \\ w_{31}^{(k+1)} & w_{32}^{(k+1)} \end{pmatrix} \begin{pmatrix} \sigma(h_1^{(k)}) \\ \sigma(h_2^{(k)}) \end{pmatrix} = \begin{pmatrix} h_1^{(k+1)} \\ h_2^{(k+1)} \\ h_3^{(k+1)} \end{pmatrix} \xrightarrow[\text{activation}]{\sigma^{(k+1)}} \begin{pmatrix} \sigma(h_1^{(k+1)}) \\ \sigma(h_2^{(k+1)}) \\ \sigma(h_3^{(k+1)}) \end{pmatrix}$$



Ndim this layer
Ndim nxt layer
$W_{11}^{(k+1)}$ $W_{12}^{(k+1)}$
$W_{21}^{(k+1)}$ $W_{22}^{(k+1)}$
$W_{31}^{(k+1)}$ $W_{32}^{(k+1)}$
Weight matrix
$\sigma(h_1^{(k)})$
$\sigma(h_2^{(k)})$
Layer input nodes
("activations")
=
$h_1^{(k+1)}$
$h_2^{(k+1)}$
$h_3^{(k+1)}$
$\sigma^{(k+1)}$
activation
$\sigma(h_1^{(k+1)})$
$\sigma(h_2^{(k+1)})$
$\sigma(h_3^{(k+1)})$
Layer output nodes
("activations")

$$\sigma\left[\begin{pmatrix} w_{11}^{(k+1)} & w_{12}^{(k+1)} \\ w_{21}^{(k+1)} & w_{22}^{(k+1)} \end{pmatrix}\begin{pmatrix} \sigma(h_1^{(k)}) \\ \sigma(h_2^{(k)}) \end{pmatrix}\right] = \sigma\begin{pmatrix} h_1^{(k+1)} \\ h_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} \sigma(h_1^{(k+1)}) \\ \sigma(h_2^{(k+1)}) \end{pmatrix}$$

$$\sigma\left[\begin{pmatrix} w_{11}^{(k)} & w_{12}^{(k)} \\ w_{21}^{(k)} & w_{22}^{(k)} \end{pmatrix}\begin{pmatrix} \sigma(h_1^{(k-1)}) \\ \sigma(h_2^{(k-1)}) \end{pmatrix}\right] = \sigma\begin{pmatrix} h_1^{(k)} \\ h_2^{(k)} \end{pmatrix} = \begin{pmatrix} \sigma(h_1^{(k)}) \\ \sigma(h_2^{(k)}) \end{pmatrix}$$

$$\sigma\left[\begin{pmatrix} w_{11}^{(k+2)} & w_{12}^{(k+2)} \\ w_{21}^{(k+2)} & w_{22}^{(k+2)} \end{pmatrix}\begin{pmatrix} \sigma(h_1^{(k+1)}) \\ \sigma(h_2^{(k+1)}) \end{pmatrix}\right] = \sigma\begin{pmatrix} h_1^{(k+2)} \\ h_2^{(k+2)} \end{pmatrix} = \begin{pmatrix} \sigma(h_1^{(k+2)}) \\ \sigma(h_2^{(k+2)}) \end{pmatrix}$$

RETINA-AI

$$\sigma\left(\begin{bmatrix} w_{11}^{(k+2)} & w_{12}^{(k+2)} \\ w_{21}^{(k+2)} & w_{22}^{(k+2)} \end{bmatrix} \begin{pmatrix} \sigma(h_1^{(k+1)}) \\ \sigma(h_2^{(k+1)}) \end{pmatrix}\right) = \sigma\begin{pmatrix} h_1^{(k+2)} \\ h_2^{(k+2)} \end{pmatrix} = \begin{pmatrix} \sigma(h_1^{(k+2)}) \\ \sigma(h_2^{(k+2)}) \end{pmatrix}$$

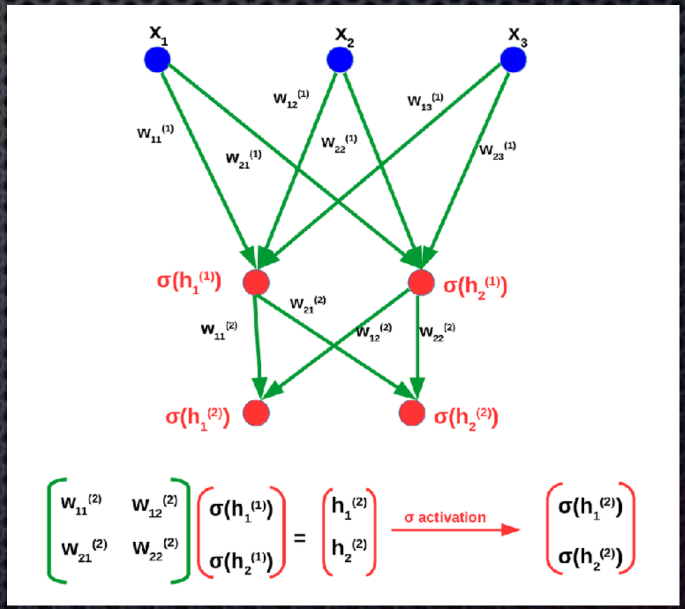
$$\sigma\left(\begin{bmatrix} w_{11}^{(k+2)} & w_{12}^{(k+2)} \\ w_{21}^{(k+2)} & w_{22}^{(k+2)} \end{bmatrix} \sigma\left(\begin{bmatrix} w_{11}^{(k+1)} & w_{12}^{(k+1)} \\ w_{21}^{(k+1)} & w_{22}^{(k+1)} \end{bmatrix} \begin{pmatrix} \sigma(h_1^{(k)}) \\ \sigma(h_2^{(k)}) \end{pmatrix}\right)\right)$$

$$\sigma\left(\begin{bmatrix} w_{11}^{(k+2)} & w_{12}^{(k+2)} \\ w_{21}^{(k+2)} & w_{22}^{(k+2)} \end{bmatrix} \sigma\left(\begin{bmatrix} w_{11}^{(k+1)} & w_{12}^{(k+1)} \\ w_{21}^{(k+1)} & w_{22}^{(k+1)} \end{bmatrix} \sigma\left(\begin{bmatrix} w_{11}^{(k)} & w_{12}^{(k)} \\ w_{21}^{(k)} & w_{22}^{(k)} \end{bmatrix} \begin{pmatrix} \sigma(h_1^{(k-1)}) \\ \sigma(h_2^{(k-1)}) \end{pmatrix}\right)\right)\right)$$

RETINA-AI

Activation Functions

ReLU, Leaky ReLU, Sigmoid, Vanishing gradients, Exploding Gradients



RETINA-AI

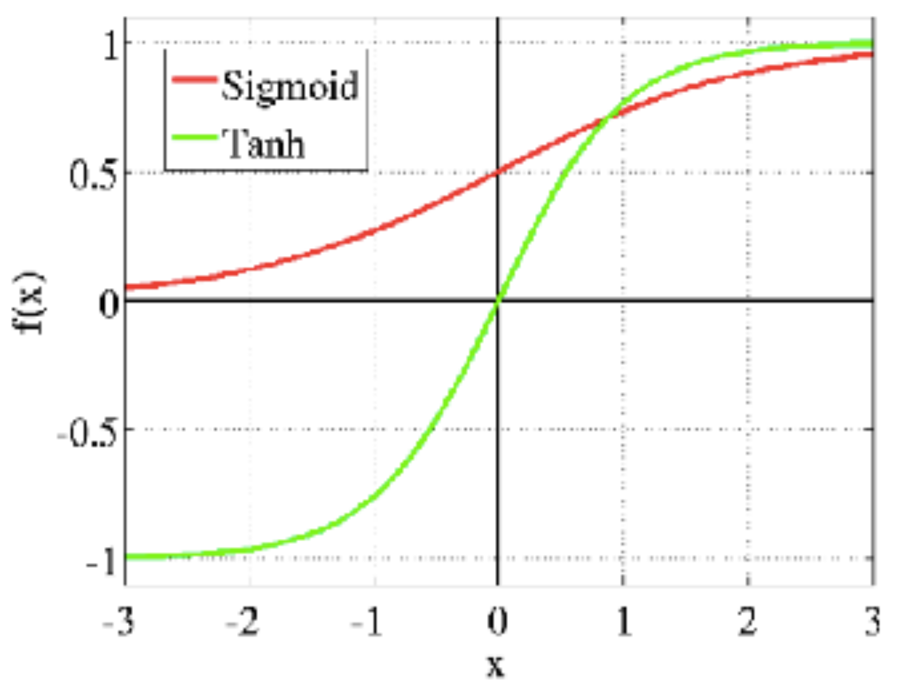


Fig: tanh v/s Logistic Sigmoid

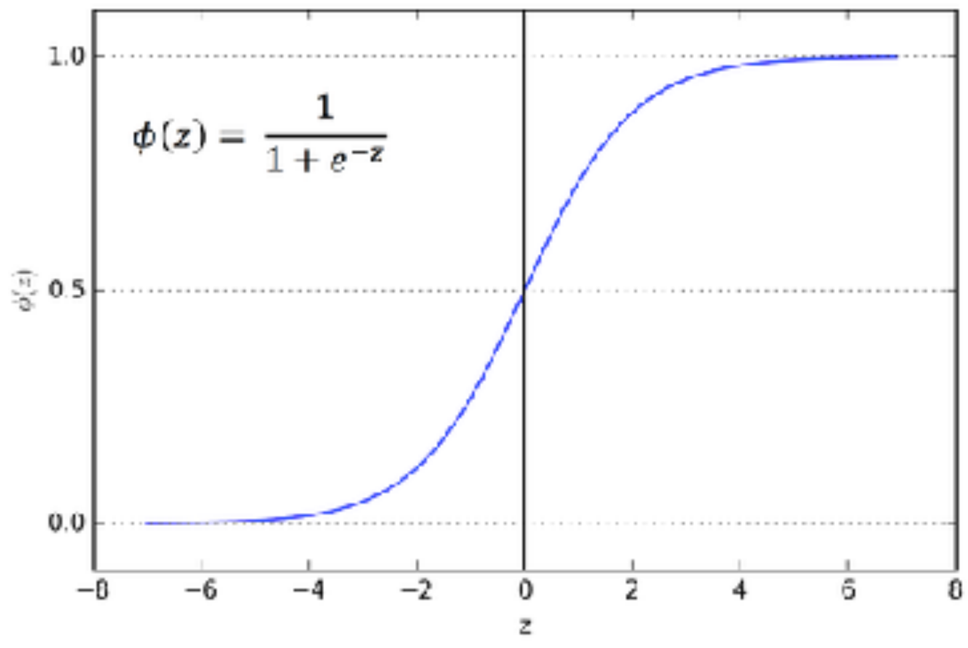


Fig: Sigmoid Function

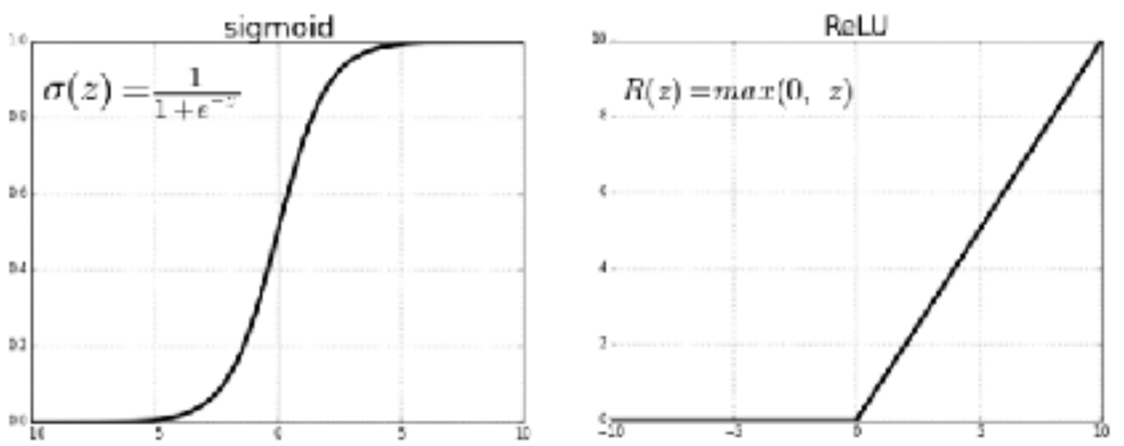


Fig: ReLU v/s Logistic Sigmoid

Sagar Sharma blog

RETINA-AI

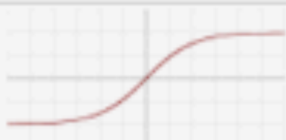
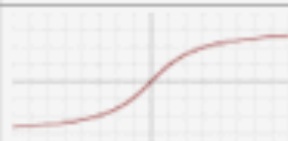
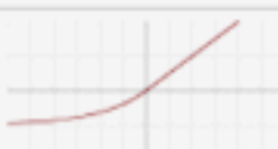
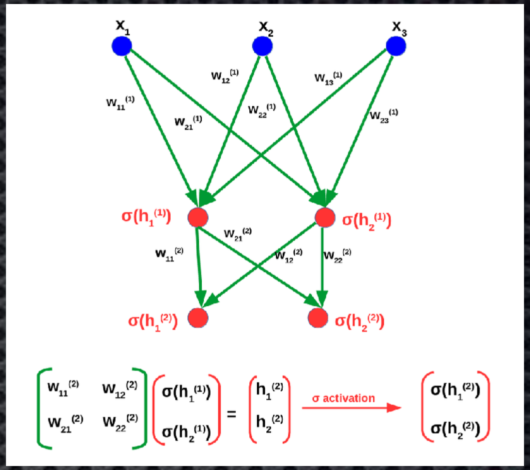
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Fig: Activation Function Cheetsheet

Sagar Sharma blog

RETINA-AI



Loss Functions

Categorical Cross-Entropy

$$-\sum_{i=1}^{C} t_i \log(\alpha(s_i))$$

Binary Cross-Entropy

$$-t_1 \log(\alpha(s_1)) - t_2 \log(\alpha(s_2)) =$$
$$-t_1 \log(\alpha(s_1)) - (1 - t_1) \log((1 - \alpha(s_1)))$$

Categorical CE, Softmax CE, Negative Log Likelihood, Multinomial Logistic Loss

$$\alpha(s)_i = \frac{e^{s_i}}{\sum_j e^{s_j}} \quad \text{and} \quad CE = -\sum_{i=1}^{C} t_i \log(\alpha(s_i))$$

RETINA-AI

One-Hot implies:

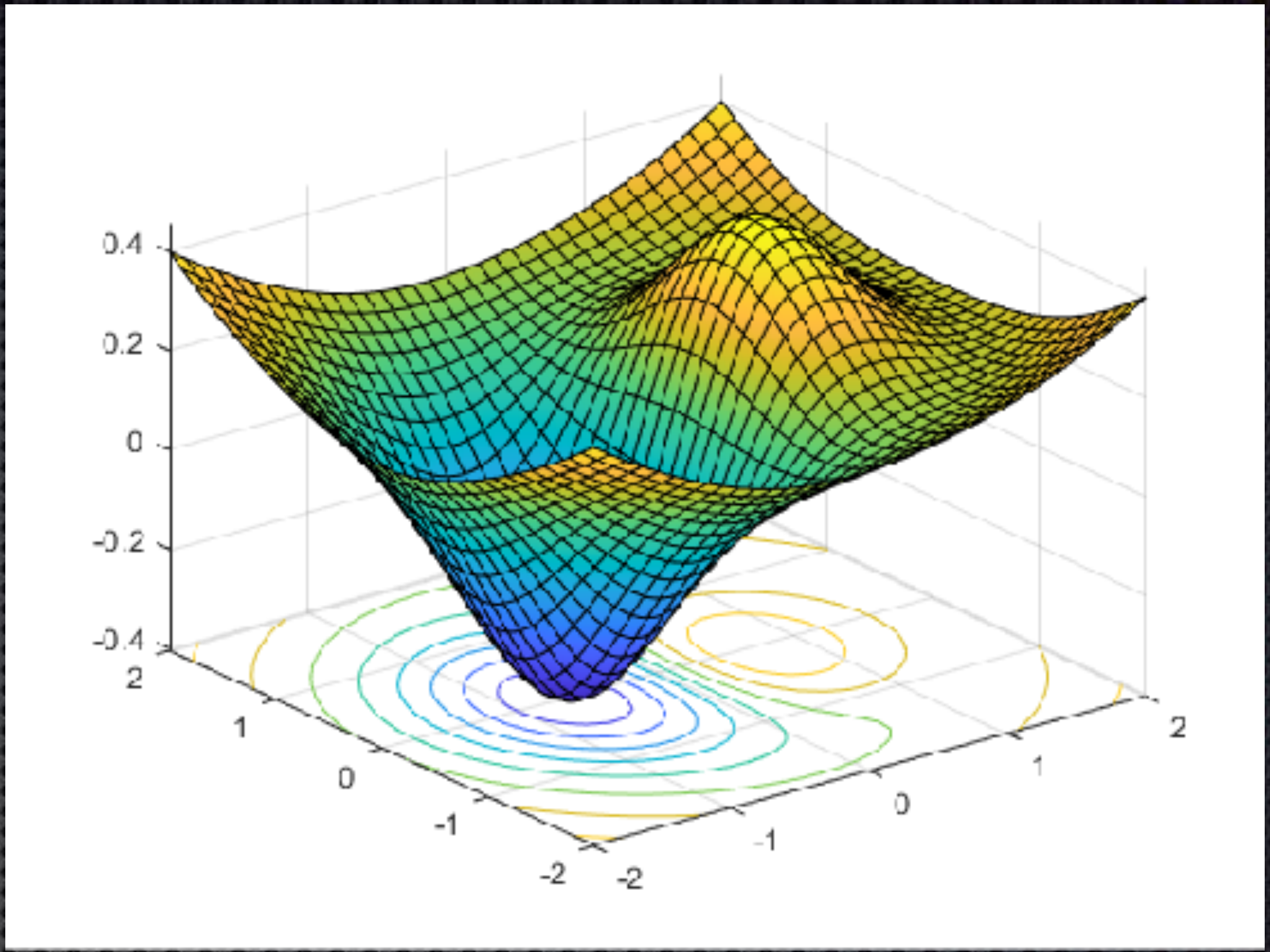
$$CE = -\log\left(\frac{e^{s_i}}{\sum_j e^{s_j}}\right)$$

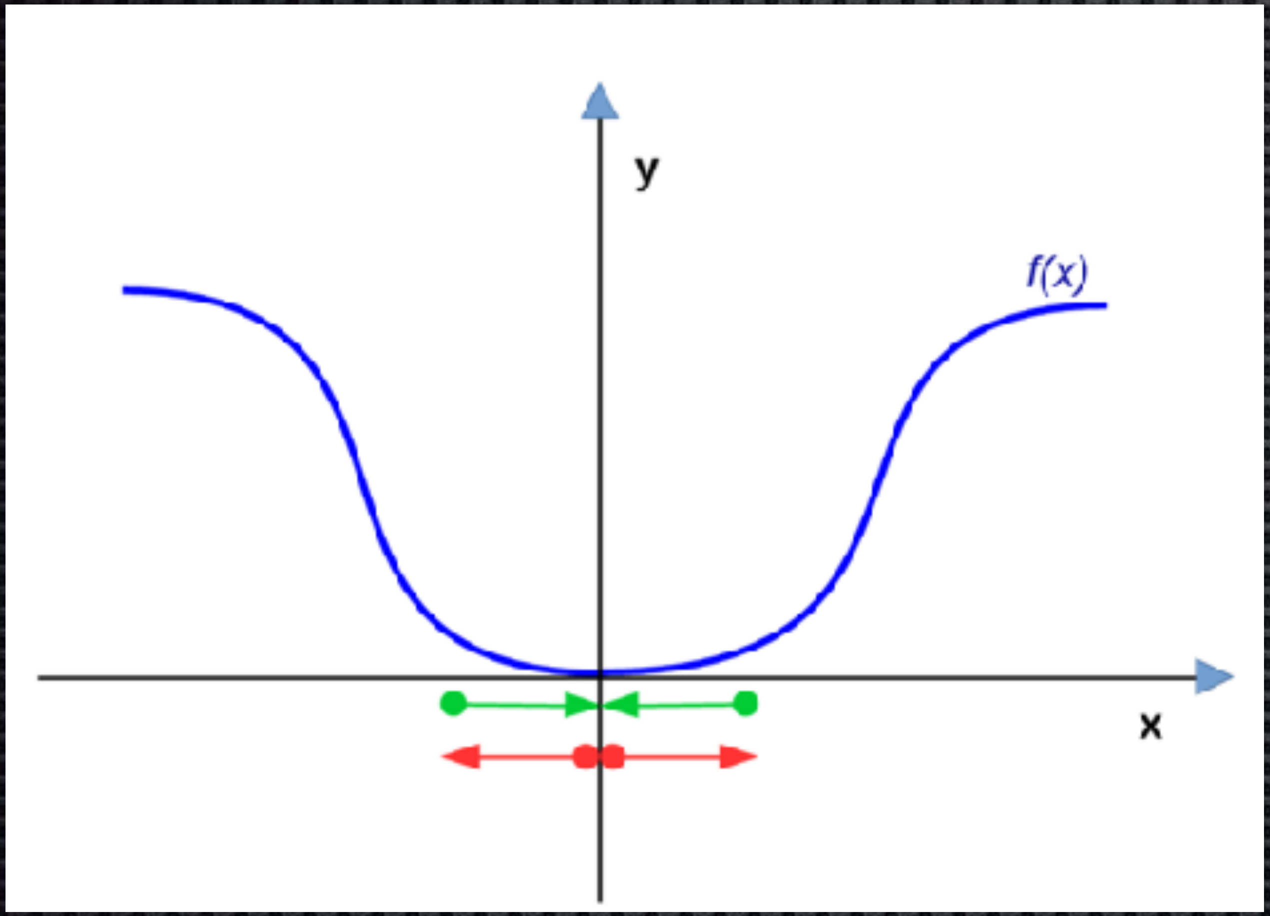
RETINA-AI

Gradient Descent

Stochastic Gradient Descent

Gradient Descent





RETINA-AI

Stochastic Gradient Descent

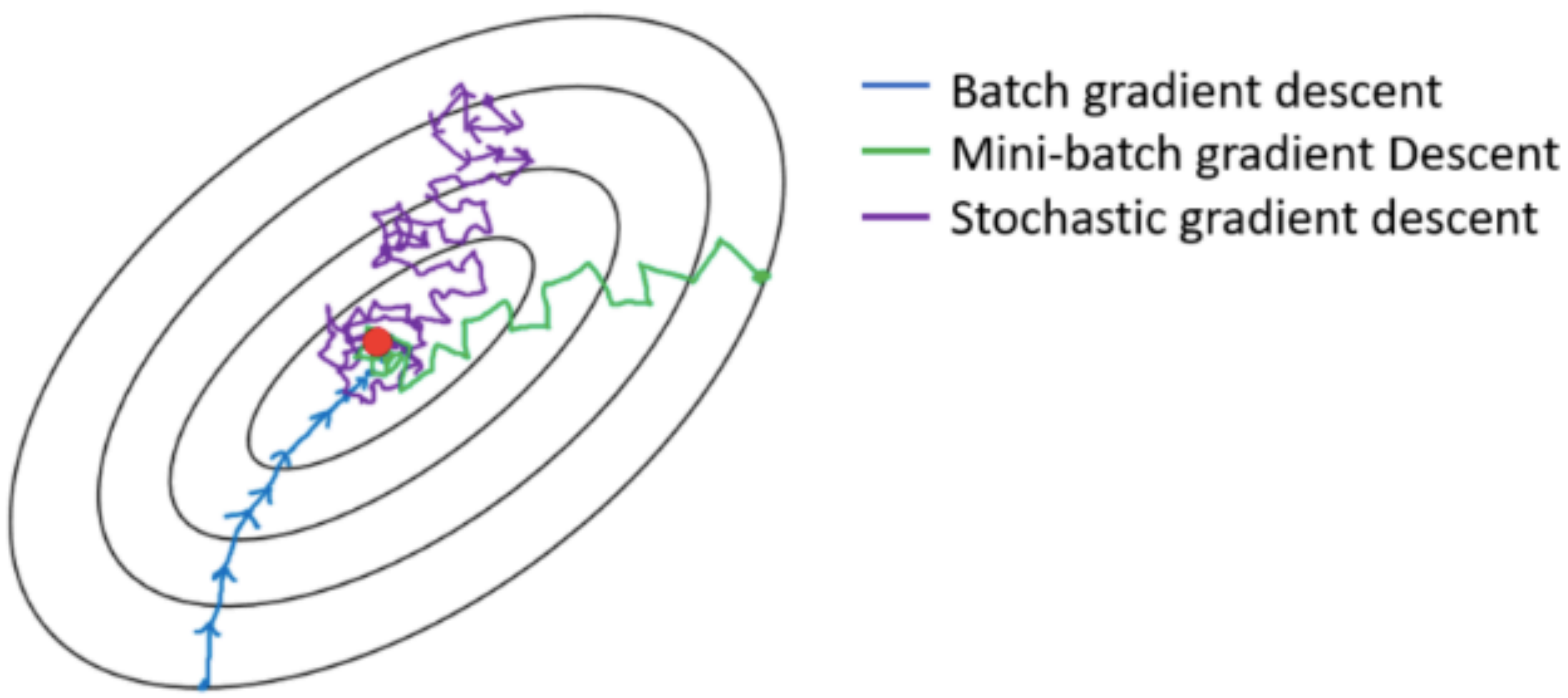


Figure 6: Gradient descent variants' trajectory towards minimum

Imad Daboura blog

Weight updates

$$w_{k+1} = w_k - \eta \frac{d\mathcal{L}}{dw_k}$$

Chain Rule

$$L = L(\sigma(w))$$

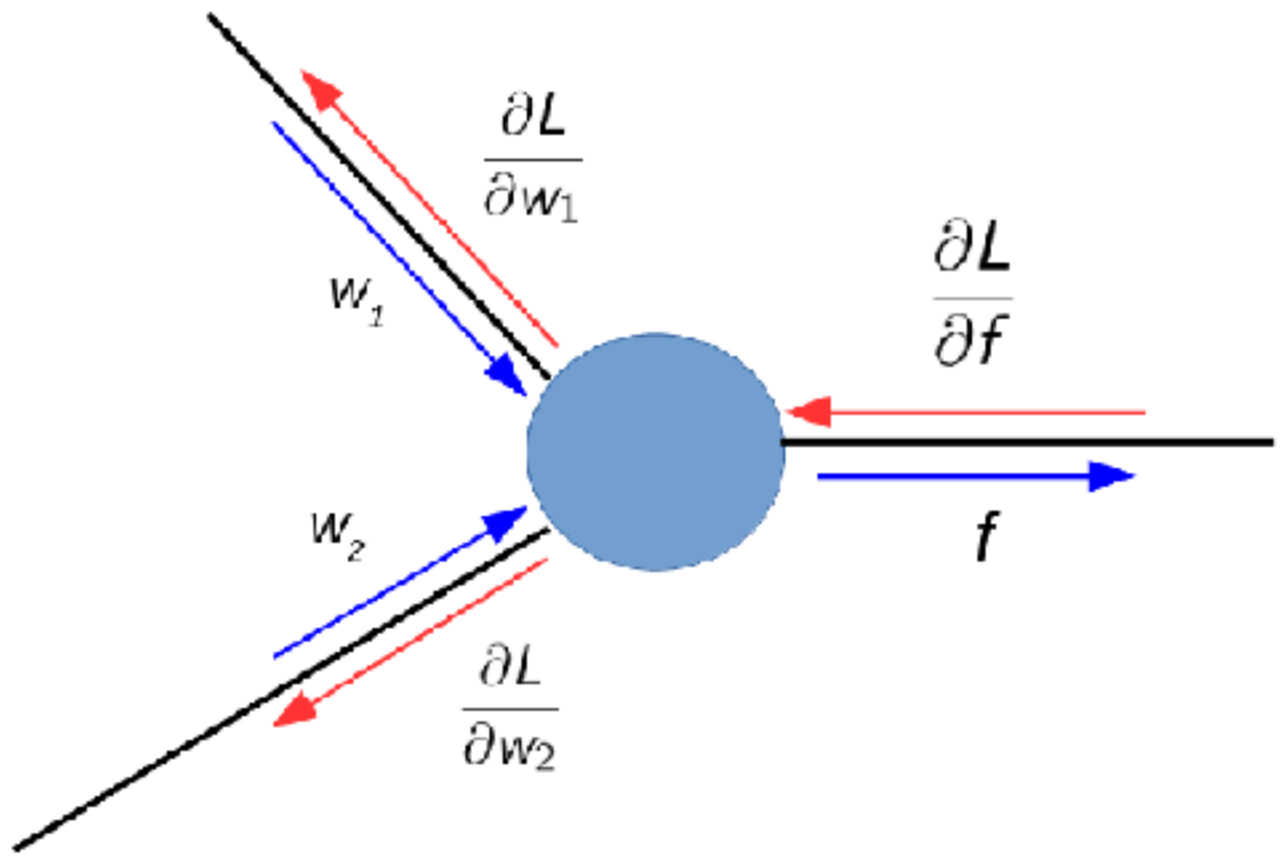
$$\frac{\partial L}{\partial w} = \frac{\partial \sigma}{\partial w}\frac{\partial L}{\partial \sigma}$$

Chain Rule

$$L = L(\gamma((\beta((\alpha(w))))))$$

$$\frac{\partial L}{\partial w} = \frac{\partial \alpha}{\partial w}\frac{\partial \beta}{\partial \alpha}\frac{\partial \gamma}{\partial \beta}\frac{\partial L}{\partial \gamma}$$

RETINA-AI



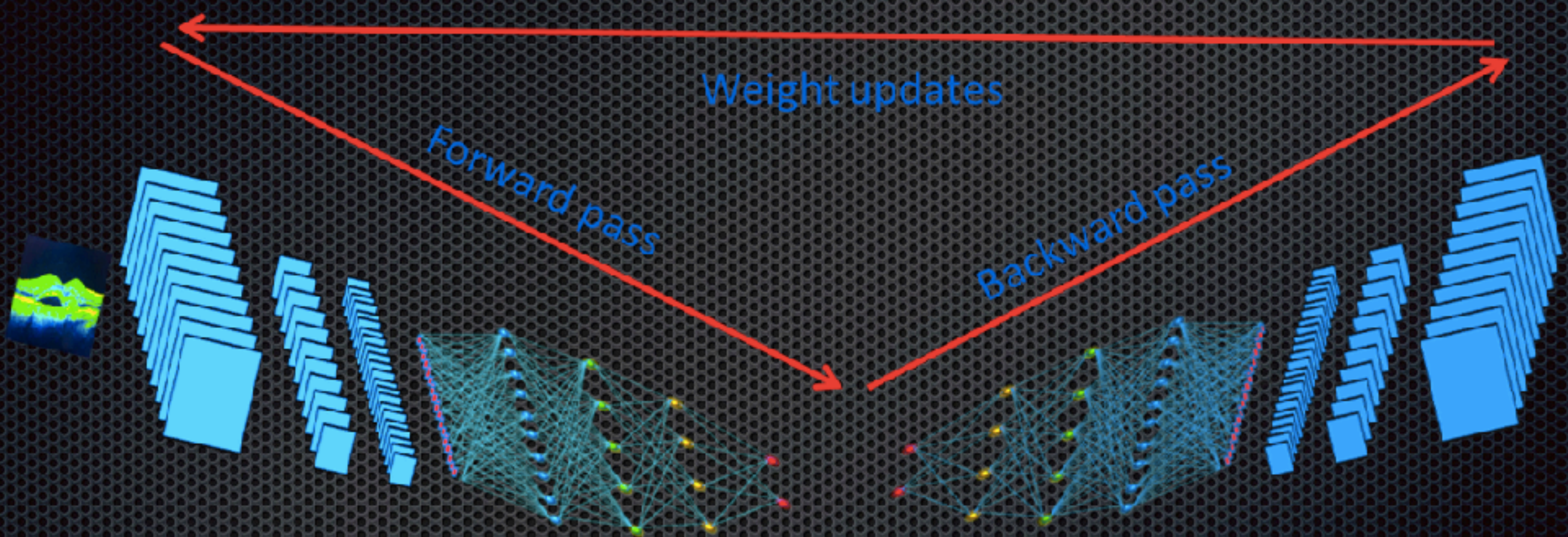
$$\frac{\partial L}{\partial w_1} = \frac{\partial f}{\partial w_1}\frac{\partial L}{\partial f}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial f}{\partial w_2}\frac{\partial L}{\partial f}$$

Weight updates

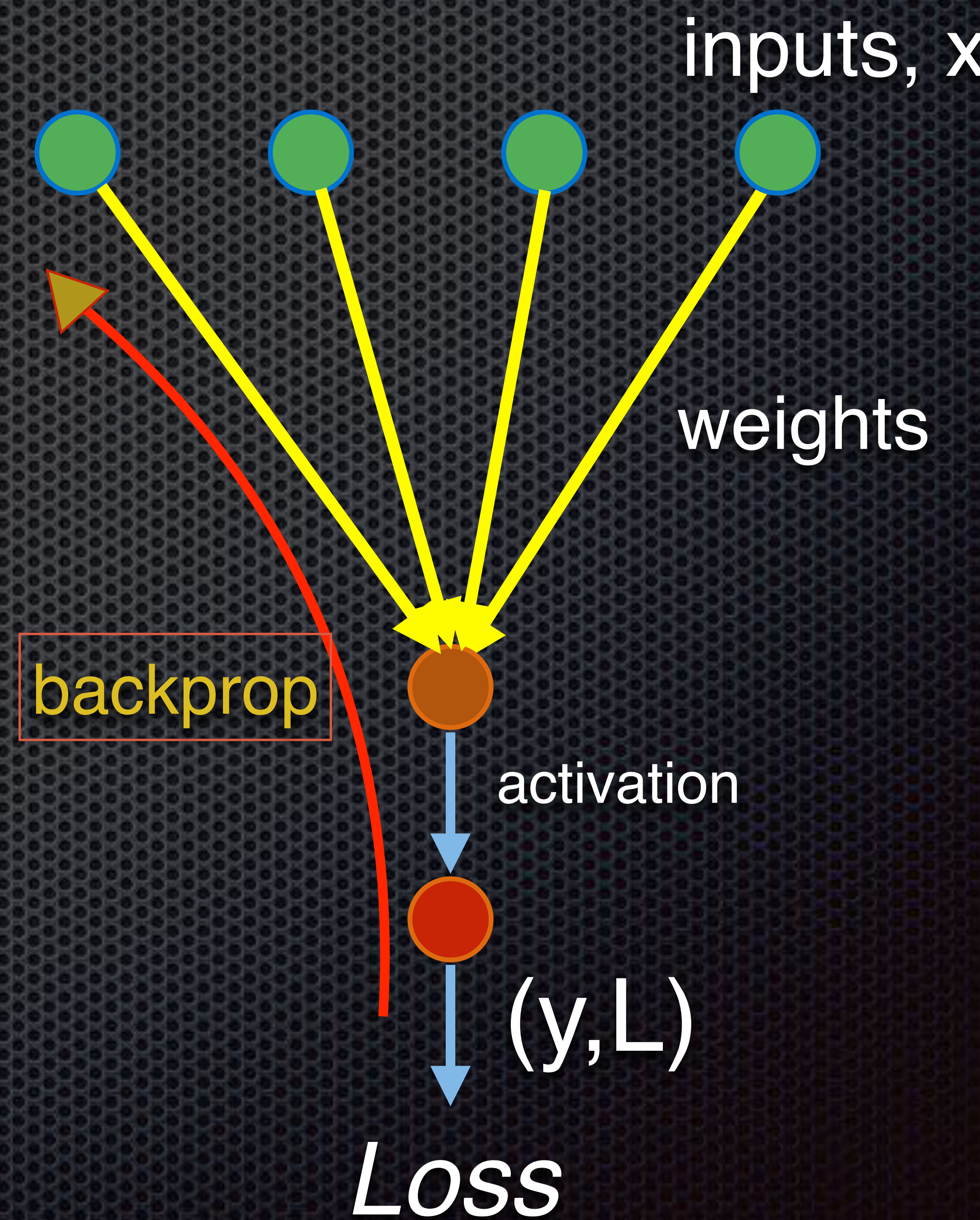
$$w_{k+1} = w_k - \eta \frac{d\mathcal{L}}{dw_k}$$

Convolutional Neural Networks



Workshop Project

- Implement in Python:
  - One Layer Neural Net
  - Sigmoid activation
  - Binary Cross Entropy



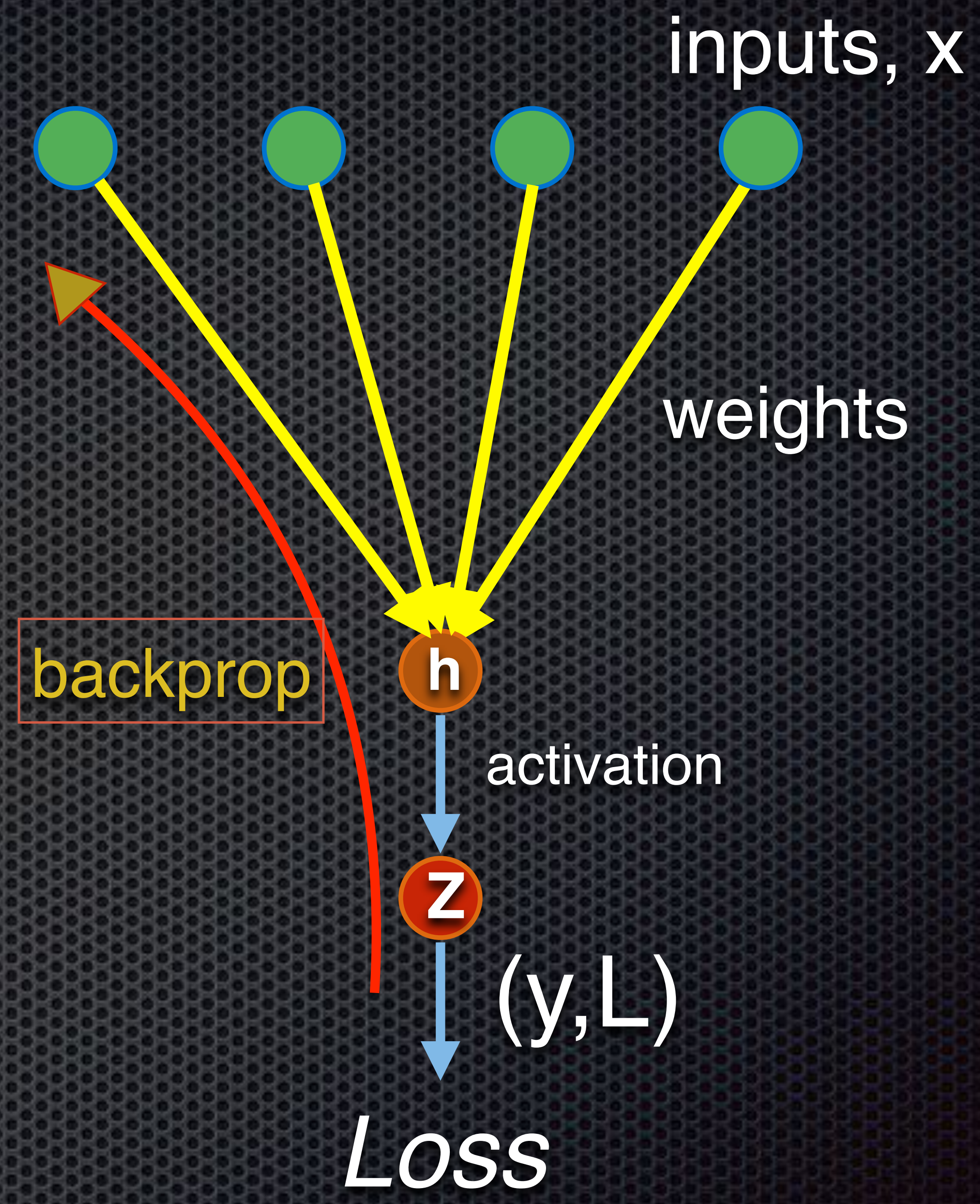
Workshop Project

- Implement in Python:
  - One Layer Neural Net
  - Sigmoid activation
  - Binary Cross Entropy

$$h = wX$$

$$z = a(h)$$

$$\text{Loss} = BCE(z,y)$$



RETINA-AI Health's Engineering Team

Acknowledgements
Olamide Oyediran
Goodness Duru
Victor Irekponor
Stanley Dukor
Udeme Udofia
Kevin Guo
Dr. Chudi Adi
Dr. Oluwatosin Smith

RETINA-AI

RETINA-AI

Artificial Intelligence DevOps Hackathon

Feb 20-22, 2020, Lagos

$7000 Prize Money