Lecture 3: Loss Functions, Optimization, and Neural Network Overview

Areej Alasiry

CIS 6217 – Computer Vision for Data Representation

College of Computer Science, King Khalid University

Outline

1. Why Loss Functions?
2. Regression Losses (L1, L2)
3. Classification Losses (Hinge, Cross-Entropy)
4. Specialized Losses in CV
5. Gradient Descent Overview
6. Batch vs Stochastic vs Mini-batch
7. Optimization Improvements (Momentum, LR schedules)
8. Neuron Model & Activation Functions
9. Feed-Forward Neural Network Structure
10. Backpropagation
11. Lab Activity (Implement a small NN)
12. Summary

Learning Outcomes

- Explain the role of loss functions in training machine learning models.
- Compare different loss functions used in computer vision tasks.
- Understand optimization techniques such as gradient descent and its variants.
- Describe the structure and operation of a basic neural network.
- Implement a simple feed-forward neural network and train it with backpropagation.

Loss Function

Measure the discrepancies between the prediction and the truth to guide training

Why Loss Functions and Optimization

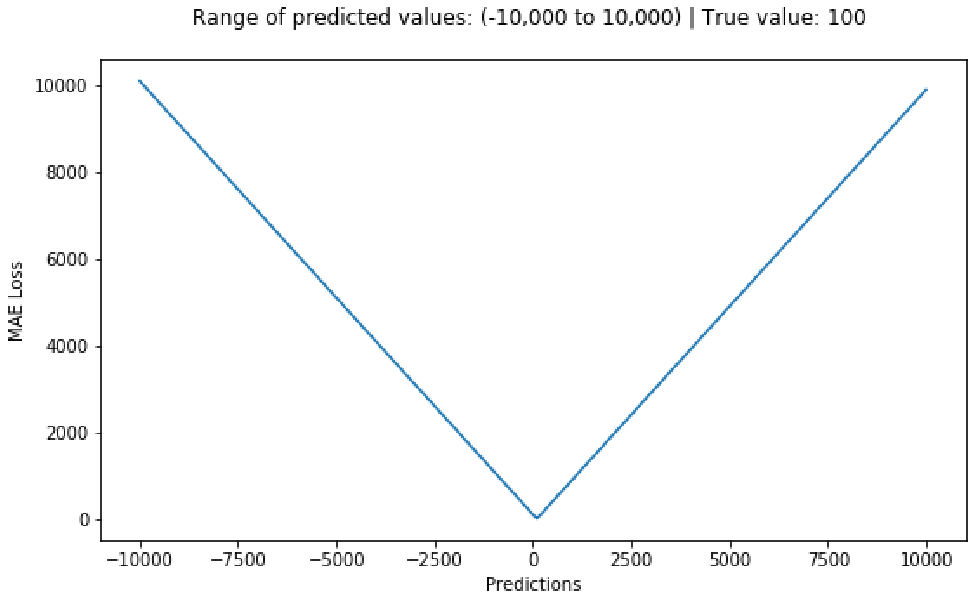
Regression Loss

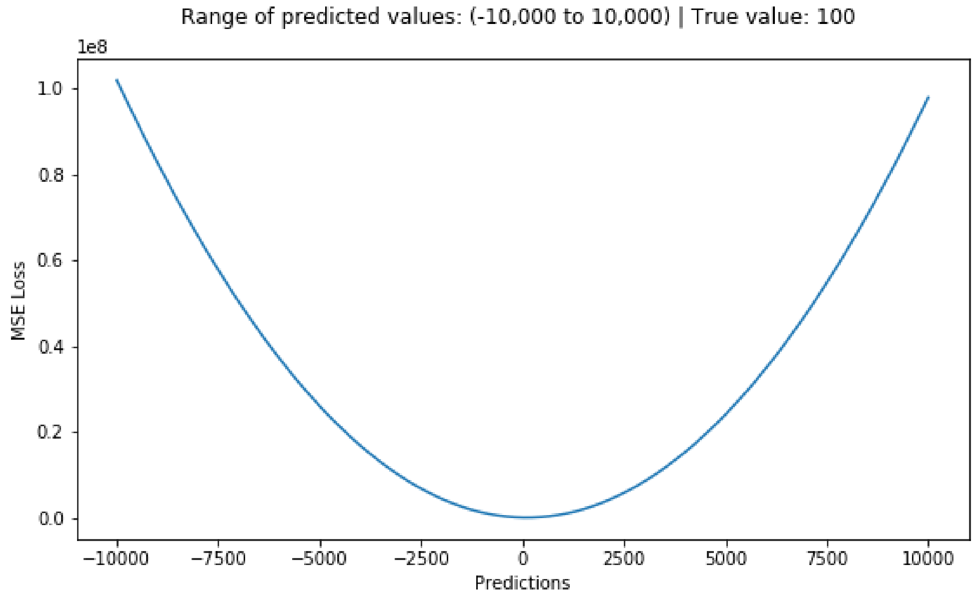
- Mean Absolute Error (MAE)

$$L1 = \sum_{i=1}^{n} \left| y_i - y_i^p \right|$$

- Mean Absolute Error (MSE)

$$L2 = \sum_{i=1}^{n} \left(y_i - y_i^p \right)^2$$





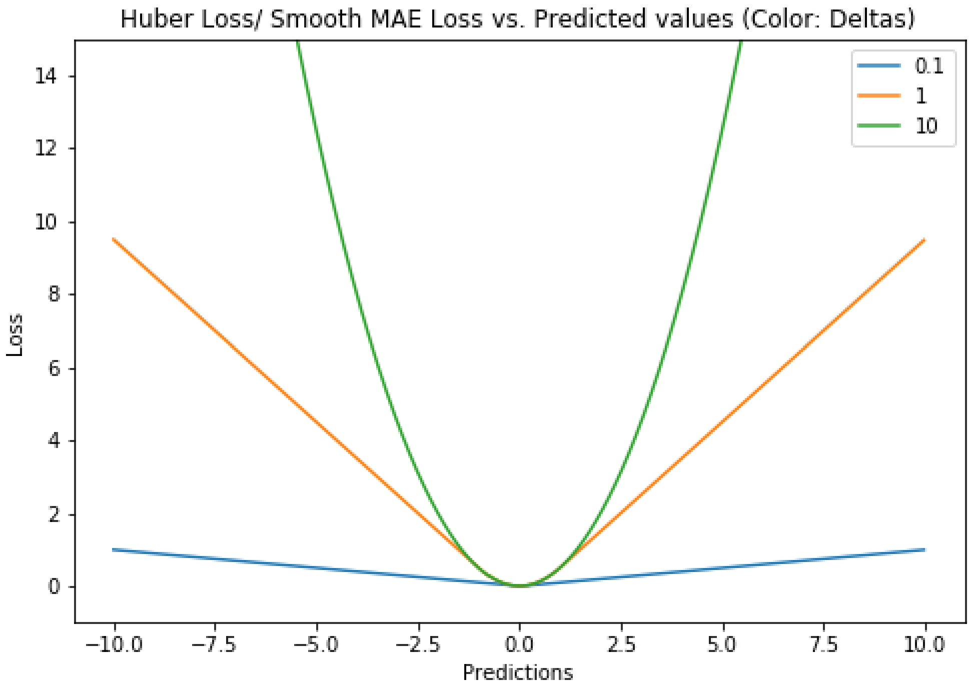
Truong, P. (2019) *Loss functions: Why, what, where or when?*, *Medium*. Available at: https://phuctrt.medium.com/loss-functions-why-what-where-or-when-189815343d3f (Accessed: 14 September 2025).

Regression Loss Cont.

- Huber or Smooth Mean Absolute Error

$$L_\delta(y, f(x)) =$$

$$\begin{cases} \frac{1}{2}(y - f(x))^2 & for\ |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & otherwise. \end{cases}$$

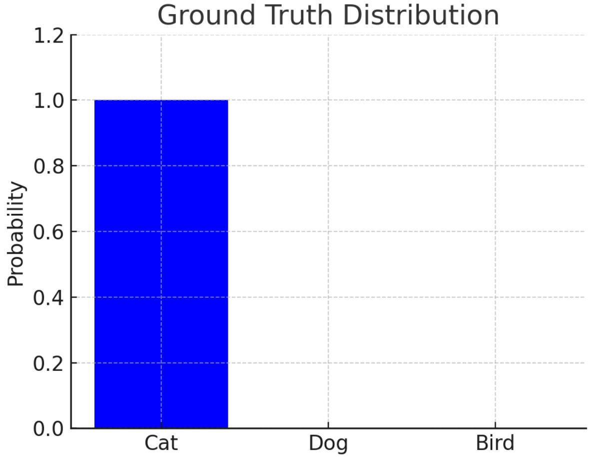


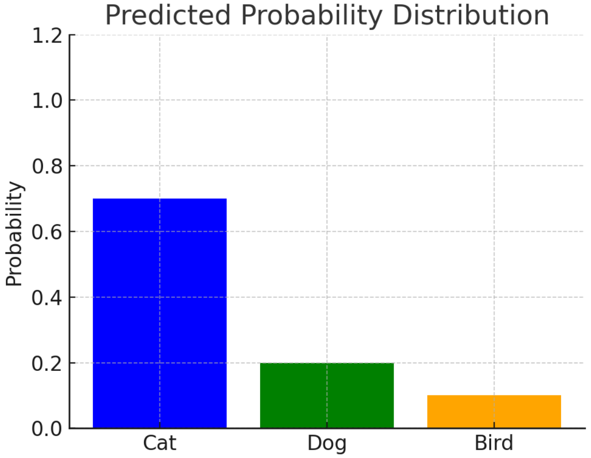
Truong, P. (2019) *Loss functions: Why, what, where or when?*, *Medium*. Available at: https://phuctrt.medium.com/loss-functions-why-what-where-or-when-189815343d3f (Accessed: 14 September 2025).

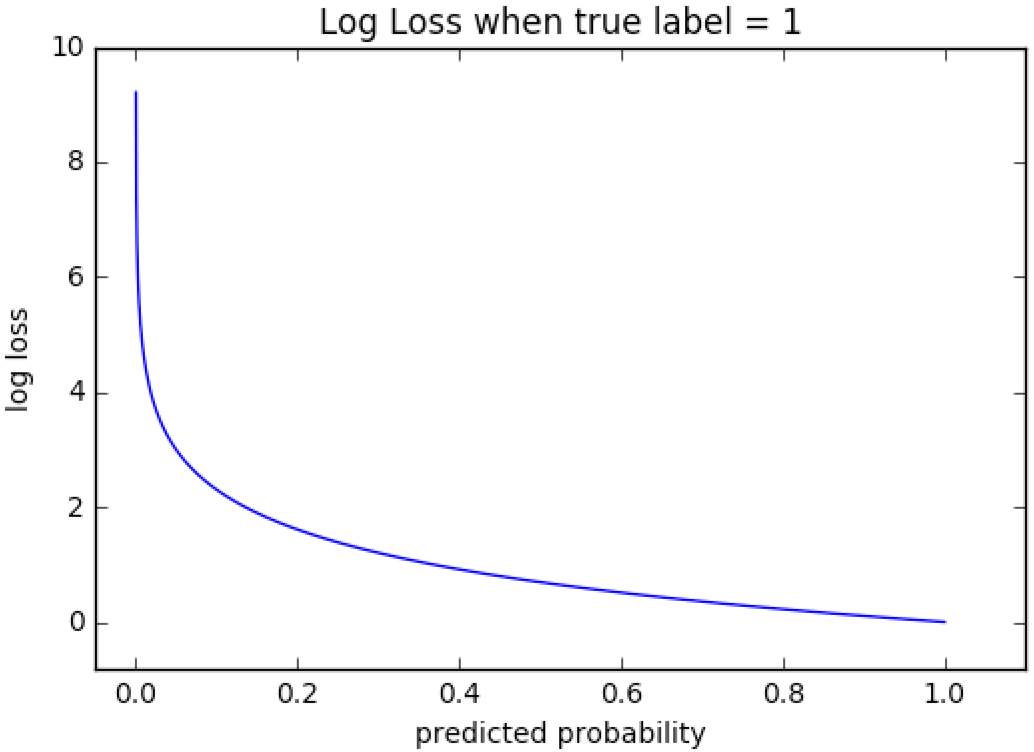
Classification Loss

- Cross-Entropy Loss (or Log Loss)

$$H(P,Q) = -\sum_{i} P(i) log Q(i)$$







Truong, P. (2019) *Loss functions: Why, what, where or when?*, *Medium*. Available at: https://phuctrt.medium.com/loss-functions-why-what-where-or-when-189815343d3f (Accessed: 14 September 2025).

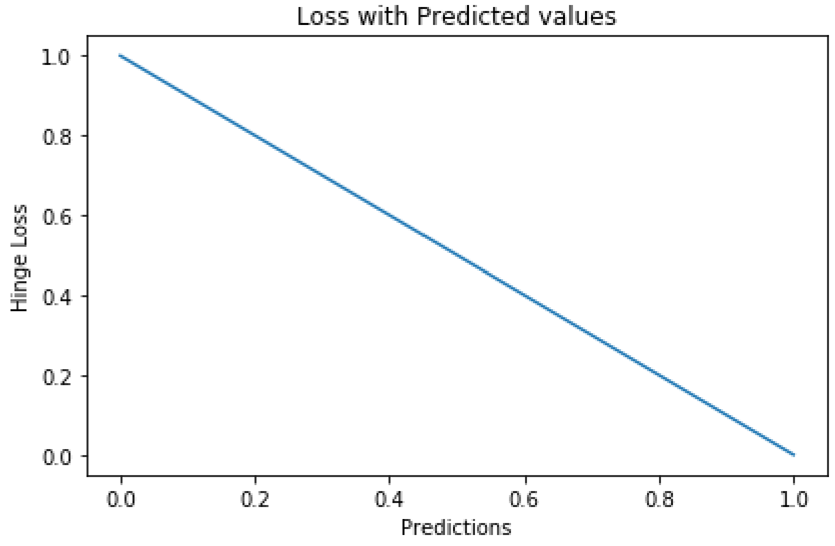
Classification Loss

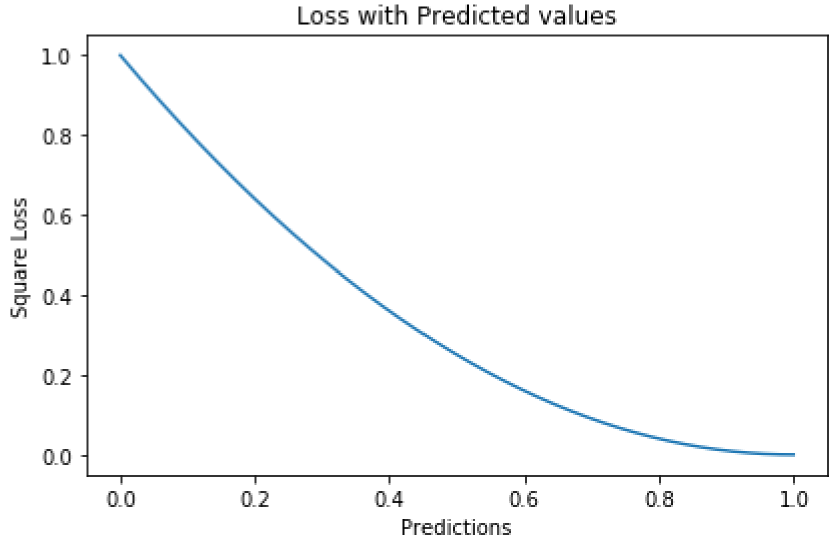
- Hinge Loss

$$J(\hat{y}, y) = \max(0,1 - y_i \cdot \hat{y})$$

- Squared Hinge Loss

$$J(\hat{y}, y) = \sum_{i=0}^{N} (\max(0,1 - y_i \cdot \hat{y})^2)$$





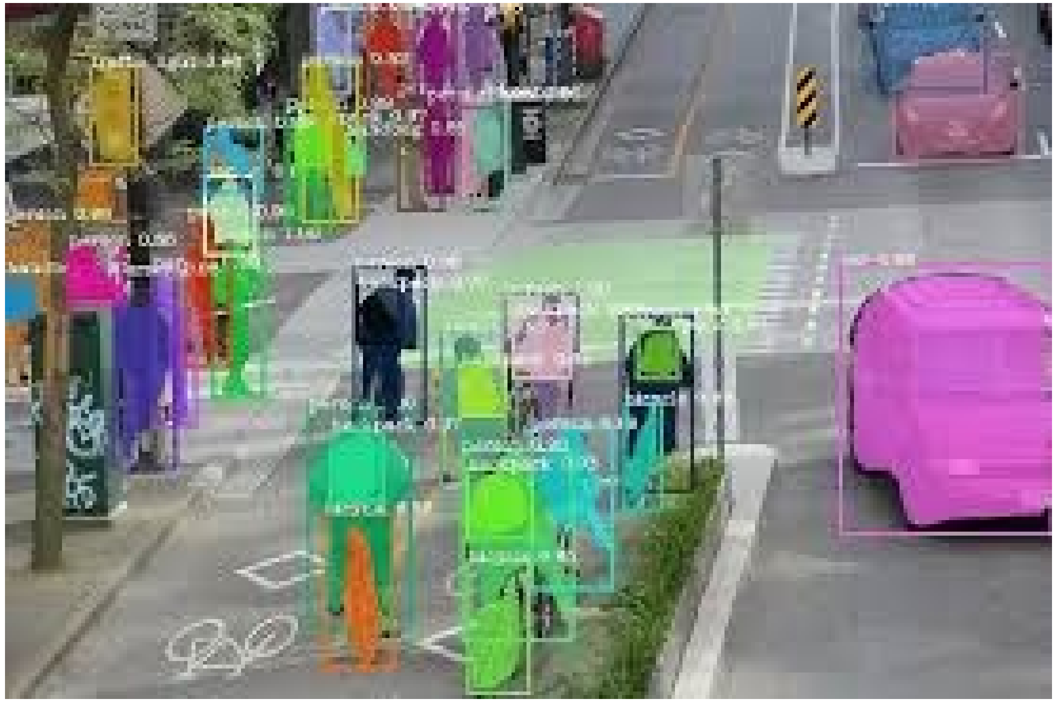
Truong, P. (2019) *Loss functions: Why, what, where or when?*, *Medium*. Available at: https://phuctrt.medium.com/loss-functions-why-what-where-or-when-189815343d3f (Accessed: 14 September 2025).

Loss in Image Segmentation

- Dice Loss = 1- Dice Coefficient

- Dice Coefficient (D)

$$D = \frac{2 \cdot |P \cap G|}{|P| + |G|}$$



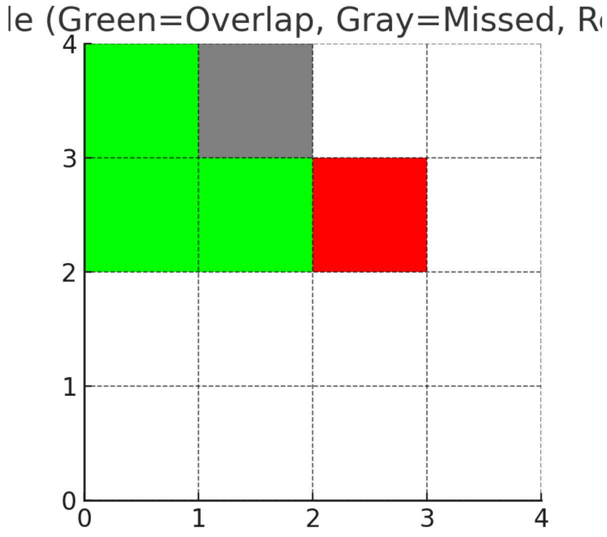
Dice Loss Illustration

1 1 0 0
1 1 0 0
0 0 0 0
0 0 0 0

G

1 0 0 0
1 1 1 0
0 0 0 0
0 0 0 0

P



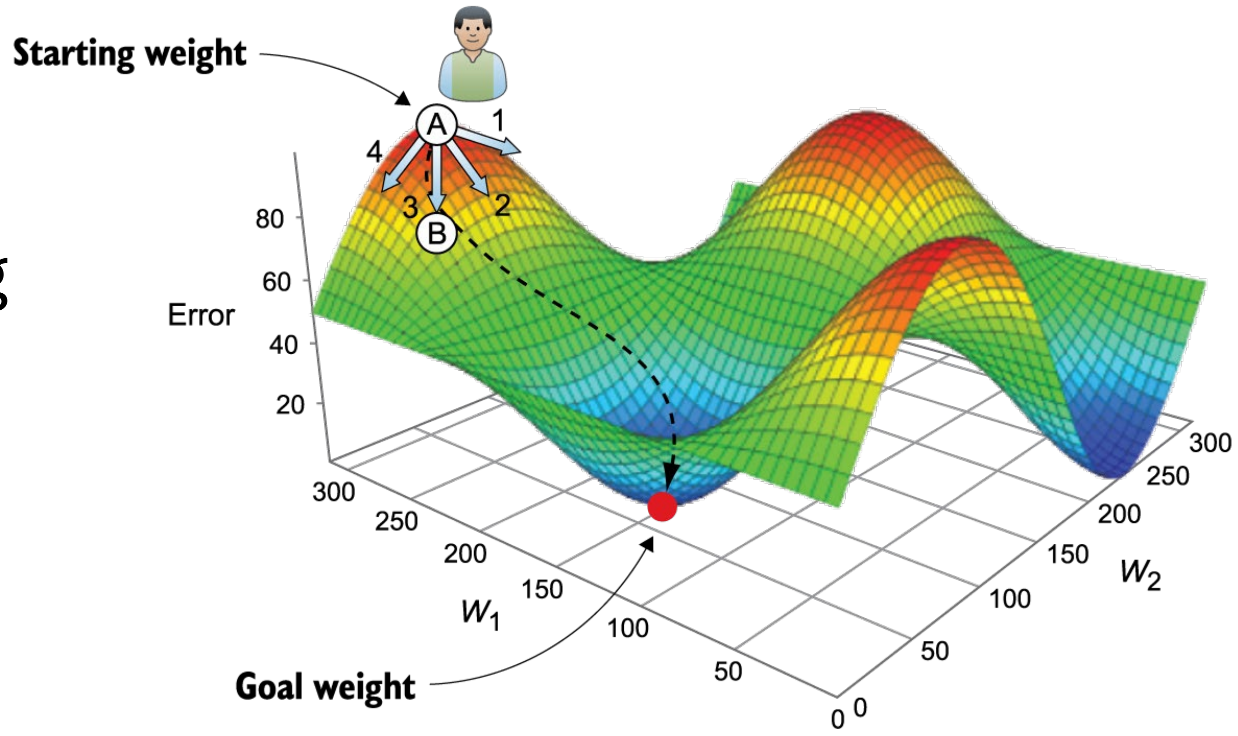
Gradient Descent

Gradually and iteratively solve a problem of function minimization .

Gradient Descent in Computer Vision

- **Gradient Descent** is an **optimization algorithm** used to minimize a loss (or cost) function by iteratively adjusting the model's parameters (weights).

$$\theta := \theta - \eta \cdot \nabla_\theta L(\theta)$$



Ref: Deep Learning for Vision Systems by Mohamed Elgendy

1. Model

$$\hat{y}_i = wx_i + b$$

2. Loss Function (MSE)

$$\mathrm{L(w, b)} = \frac{1}{n}\sum_{1}^{n}(y_i - (wx_i + b))^2$$

3. Derivatives

Derivative wrt **w**:

$$\frac{\partial L}{\partial w} = \frac{-2}{n}\sum_{i=1}^{n} x_i\,(y_i - (wx_i + b))$$

Derivative wrt **b**:

$$\frac{\partial L}{\partial b} = \frac{-2}{n}\sum_{i=1}^{n}(\,y_i - (wx_i + b))$$

4. Update Rules (Gradient Descent)

$$w := w - \eta \cdot \frac{\partial L}{\partial w}$$

$$b := b - \eta \cdot \frac{\partial L}{\partial b}$$

5. Walkthrough Example

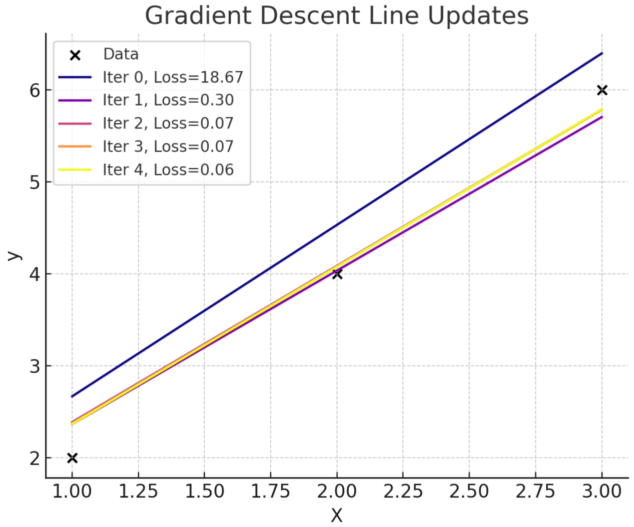
Dataset:

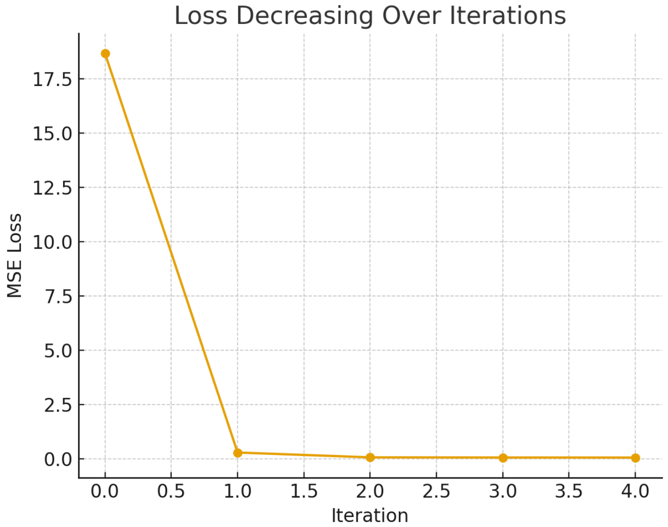
$$X = [1, 2, 3], y = [2, 4, 6]$$

Learning rate: $\eta = 0.1.$

Initial: $w = 0, b = 0.$

Iteration	w	b	Loss
0	1.866667	0.8	18.66667
1	1.671111	0.693333	0.296296
2	1.700741	0.686222	0.073376
3	1.70556	0.668681	0.067396
4	1.712898	0.652721	0.064165





Variations of Gradient Descent

- Batch Gradient Descent

$$w := w - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla_w L(x_i, y_i)$$

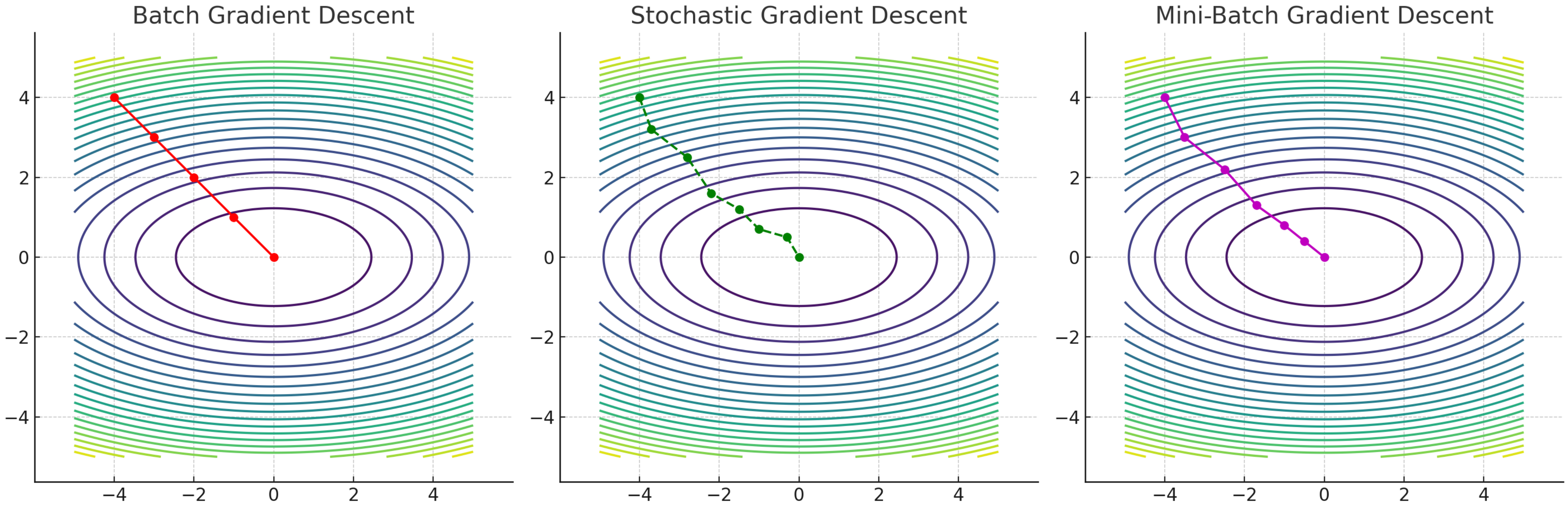
- Stochastic Gradient Descent

$$w := w - \eta \cdot \nabla_w L(x_i, y_i) \qquad (for\ a\ random\ sample\ i)$$

- Mini Batch Gradient Descent

$$w := w - \eta \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla_w L(x_i, y_i) \qquad where\ m = batch\ size$$

Illustration of Gradient Descent Variations



Advanced Variations of Gradient Descent

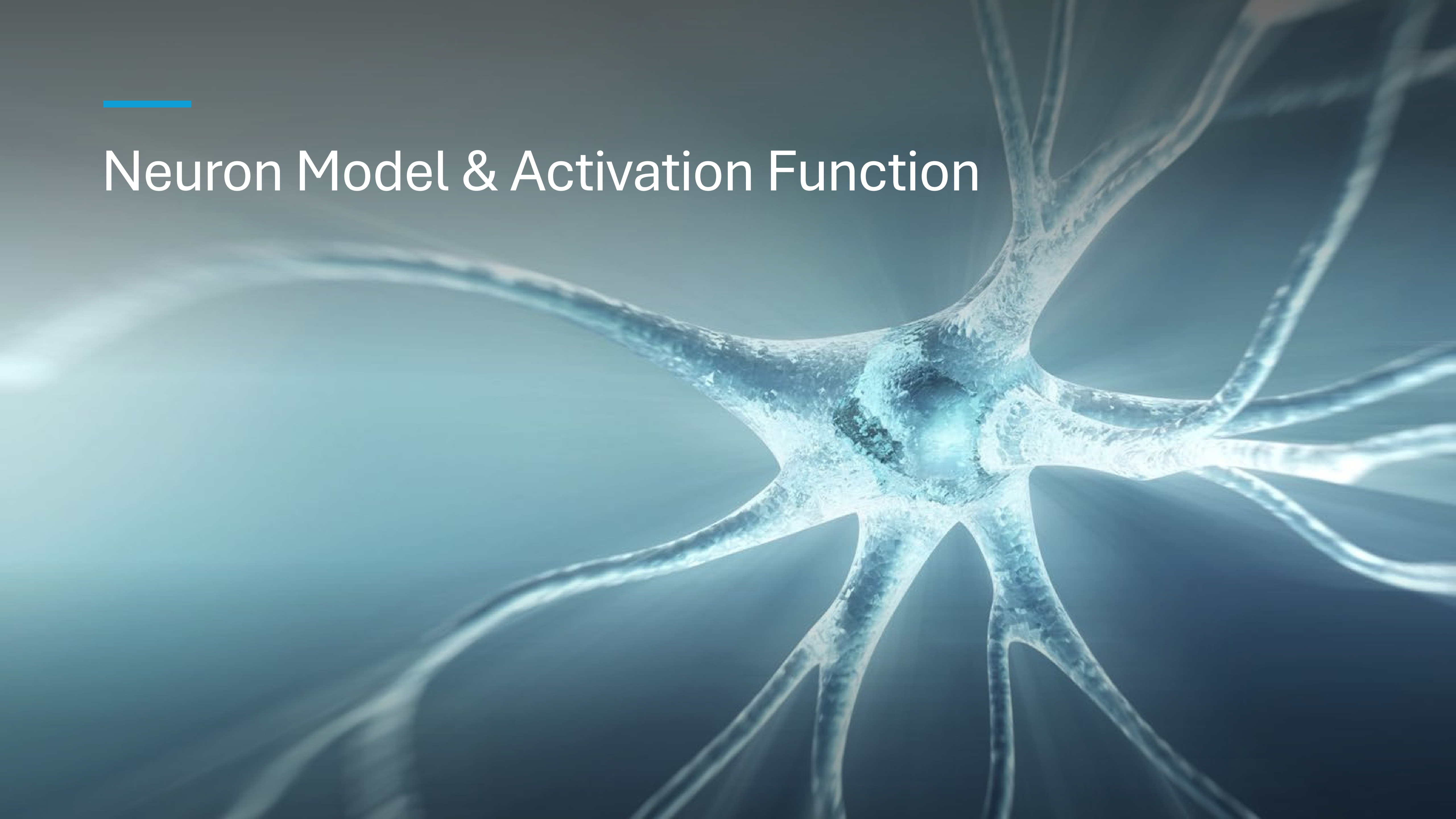
- Momentum: Adds a **velocity term** to smooth updates.

$$v := \beta v + (1-\beta)\nabla_w L, w := w - \eta v$$

- **RMSProp:** Scales learning rate for each parameter based on past gradients.

- **Adam (Adaptive Moment Estimation):** Combines **Momentum + RMSProp**.

Neuron Model & Activation Function



Neuron Model

Input features: $x_1, x_2, \ldots, x_n$

Weights: $w_1, w_2, \ldots, w_n)$

Bias: b

- Weighted sum:

$$z = \sum_{i=1}^{n} w_i\, x_i + b$$

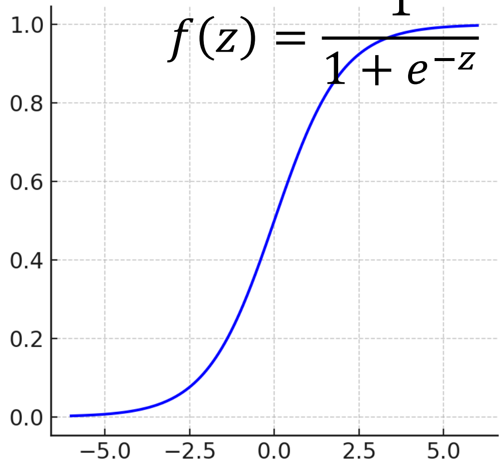
- Activation function:

$$a = f(z)$$

Activation Functions

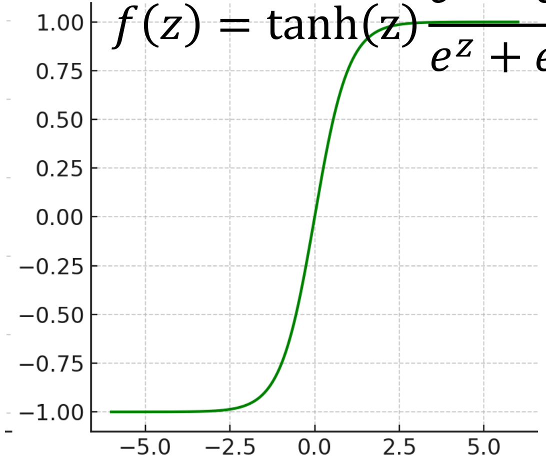
1. Sigmoid

$$f(z) = \frac{1}{1 + e^{-z}}$$



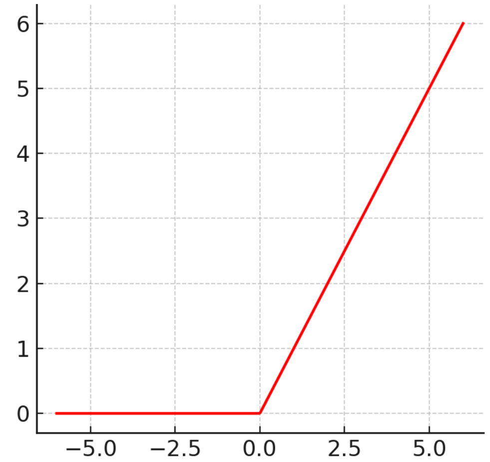
2. Tanh

$$f(z) = \tanh(z) \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



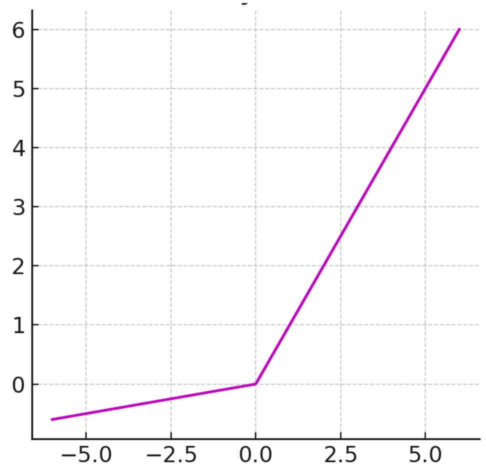
3. ReLU (Rectified Linear Unit)

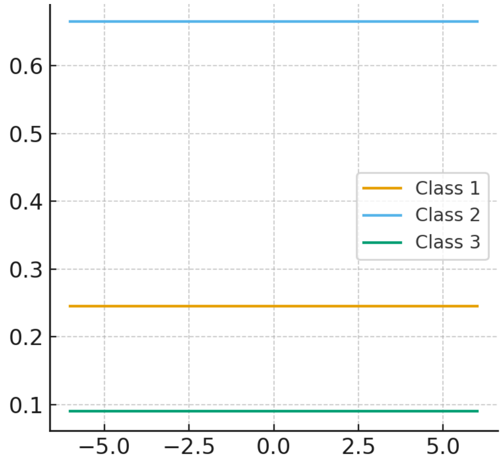
$$f(z) = \max(0, z)$$



4. Leaky ReLU

$$f(z) = \begin{cases} z & z \geq 0 \\ \alpha z & z < 0 \end{cases}$$





5. Softmax

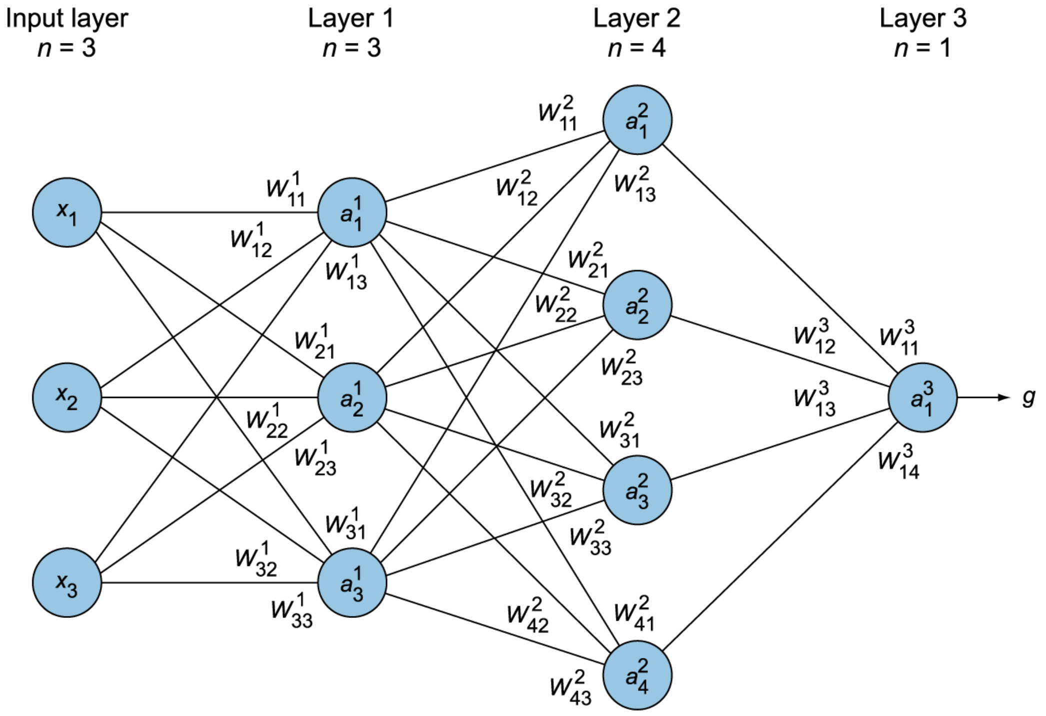
$$f(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Feed-Forward Neural Network

- A **feed-forward neural network** is the simplest type of artificial neural network, where information flows **in one direction** — from input → hidden layers → output, without cycles or feedback loops.

- **Structure**
  - **Input Layer:** Raw features (e.g., pixels of an image).
  - **Hidden Layers:** Neurons with weights, biases, and activations that transform input into abstract representations.
  - **Output Layer:** Produces final predictions (e.g., class probabilities with softmax).

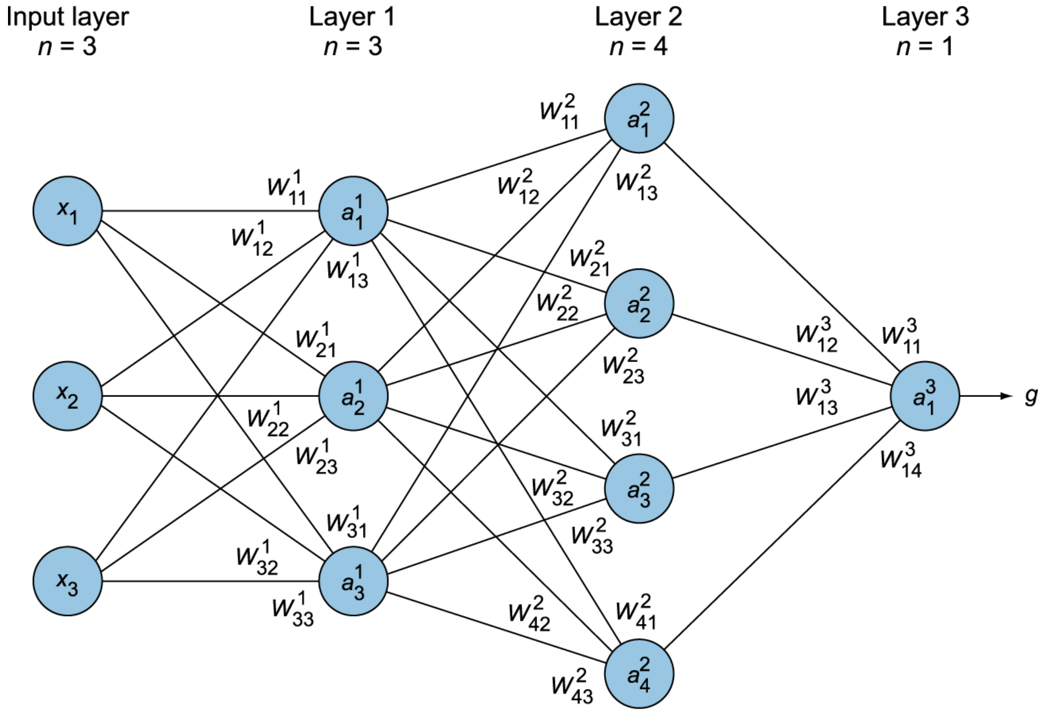
Forward Pass



Ref: Deep Learning for Vision Systems by Mohamed Elgendy

- **Computation Flow**
- Take inputs: $x_1, x_2, \ldots, x_n$.
- Multiply by weights and add biases:
$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$
- Apply activation function:
$$a^{(l)} = f\left(z^{(l)}\right)$$
- Repeat through hidden layers until output.

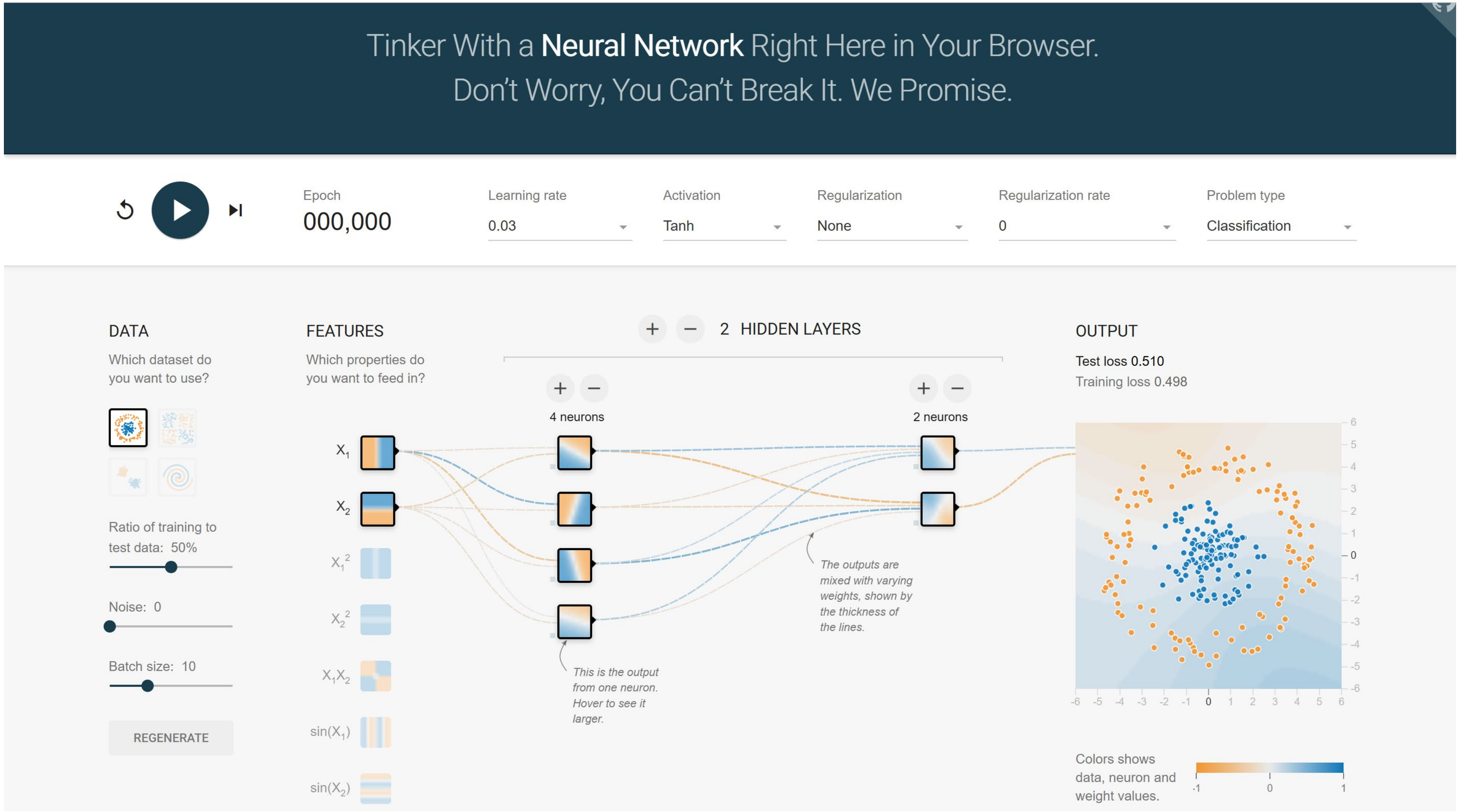
Calculation



$$\hat{y} = \sigma \underbrace{\overset{W^{(3)}}{\begin{bmatrix} w^3_{11} & w^3_{12} & w^3_{13} & w^3_{14} \end{bmatrix}}}_{\text{Layer 3}} \cdot \sigma \underbrace{\overset{W^{(2)}}{\begin{bmatrix} w^2_{11} & w^2_{12} & w^2_{13} \\ w^2_{21} & w^2_{22} & w^2_{23} \\ w^2_{31} & w^2_{32} & w^2_{33} \\ w^2_{41} & w^2_{42} & w^2_{43} \end{bmatrix}}}_{\text{Layer 2}} \cdot \sigma \underbrace{\overset{W^{(1)}}{\begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ w^1_{31} & w^1_{32} & w^1_{33} \end{bmatrix}}}_{\text{Layer 1}} \underbrace{\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}}_{\text{Input vector}}$$

Ref: Deep Learning for Vision Systems by Mohamed Elgendy

Play around with NN



https://tinyurl.com/2rekcckc

Backpropagation

Forward Pass

Input flows through the network.

Compute predictions $\hat{y}$.

Compute loss $L(\hat{y}, y)$.

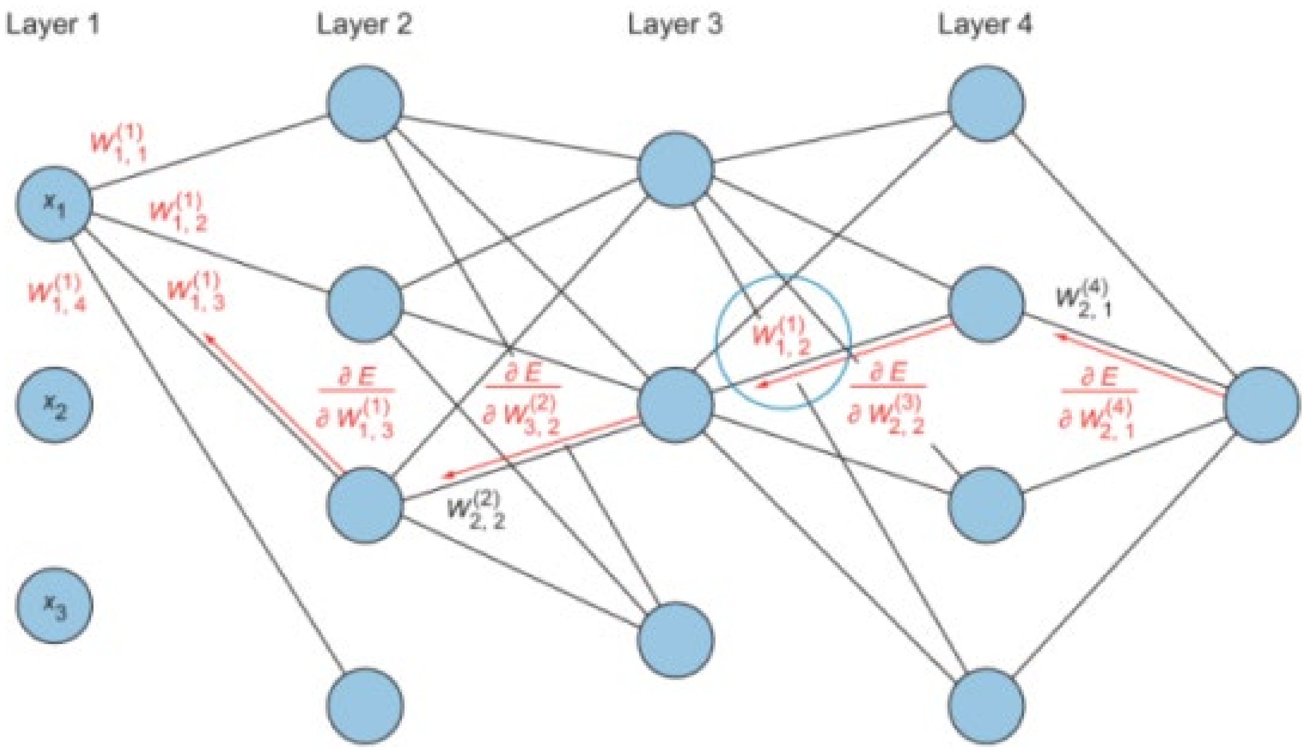
Backward Pass (Backpropagation)

Compute gradient of the loss wrt outputs of the last layer.

Apply **chain rule** layer by layer:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

Update weights using gradient descent.



Ref: Deep Learning for Vision Systems by Mohamed Elgendy

Summary

- Loss Functions: quantify error (L1/L2, Cross-Entropy, Dice).
- Optimization: Gradient Descent + its variants (Batch, SGD, Mini-Batch); advanced optimizers (Momentum, RMSProp, Adam).
- Neuron Model: weighted sum + bias + activation
- Feed-Forward NN: Input → Hidden Layers → Output, universal approximators.
- Backpropagation: chain rule-based algorithm to update weights and minimize loss.

References

- Computer Vision: A Modern Approach – Forsyth & Ponce (2010)
- Extra: Deep Learning for Vision Systems by Mohamed Elgendy