Calculus Used in Deep Learning

Intuition, Theory, and Worked Examples

Praveen Kumar Chandaliya

Department of Artificial Intelligence
SVNIT Surat

January 9, 2026

Outline

1. Why Calculus is Needed in Deep Learning
2. Derivative: Basic Concept
3. Neuron Model and Derivatives
4. Loss Function and Optimization
5. Chain Rule and Backpropagation
6. Taylor Series Definition
7. Step-by-Step Example
8. First and Second Order Approximations
9. Numerical Taylor Approximation
10. Activation Functions and Derivatives
11. Summary

Why Calculus?

- Deep learning models learn by **minimizing a loss function**
- Learning requires knowing:
  - How output changes with weights
  - Which direction reduces the error
- Calculus provides:
  - Derivatives
  - Gradients
  - Optimization methods

Core idea:

$$\text{Learning} = \text{Gradient-based optimization}$$

Derivative: Intuition

Definition

The derivative measures the rate of change of a function with respect to its input. A derivative tells us how much the output changes when the input changes.

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

- Large derivative $\rightarrow$ output changes rapidly
- Small derivative $\rightarrow$ output changes slowly

Single Neuron Model

A neuron computes:

$$z = wx + b$$

where:

- w – weight
- x – input
- b – bias

Derivative of Neuron Output

Derivative of z with respect to w:

$$\frac{\partial z}{\partial w} = x$$

Interpretation:

- Large input $\rightarrow$ strong weight update
- Small input $\rightarrow$ weak learning signal

Partial Derivatives: Multiple Inputs and Weights

Consider a neuron with multiple inputs:

$$z = w_1 x_1 + w_2 x_2. \tag{1}$$

The partial derivatives of z with respect to each weight are given by:

$$\frac{\partial z}{\partial w_1} = x_1, \qquad \frac{\partial z}{\partial w_2} = x_2. \tag{2}$$

Interpretation: Each weight is updated independently and the magnitude of the update is proportional to its corresponding input feature.

Loss Function

Loss quantifies prediction error.
Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$$

- y – true label
- $\hat{y}$ – predicted output

Derivative of Loss

Derivative of loss w.r.t. prediction:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

Example:

$$y = 1, \quad \hat{y} = 0.6$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -0.4$$

Meaning: Prediction must increase.

Chain Rule

If:

$$y = f(g(x))$$

that is, y depends on g and g depends on x then x, then

$$\frac{dy}{dx} = \frac{dy}{dg} \cdot \frac{dg}{dx}$$

Interpretation (Intuition): Think of the flow

$$x \longrightarrow g(x) \longrightarrow y$$

A small change in x affects $g(x)$, and that change in $g(x)$ affects y. Therefore, the total rate of change is the product of the two rates:

$$\frac{dy}{dx} = \frac{dy}{dg} \cdot \frac{dg}{dx}.$$

Backpropagation = repeated chain rule

Examples

i)

$$y = \sin(x^2).$$

Outer function: $f(u) = \sin u$, Inner function: $g(x) = x^2$.
Then, by the chain rule,

$$\frac{dy}{dx} = \cos(x^2) \cdot 2x.$$

ii)

$$y = e^{3x} \quad \Rightarrow \quad \frac{dy}{dx} = 3e^{3x}.$$

iii)

$$y = e^{x^2} \quad \Rightarrow \quad \frac{dy}{dx} = 2x\, e^{x^2}.$$

iv)

$$y = e^{e^x} \quad \Rightarrow \quad \frac{dy}{dx} = e^{e^x} \cdot e^x.$$

Backpropagation Example

Neuron with activation:

$$z = wx + b, \quad a = \sigma(z)$$

Loss:

$$\mathcal{L}(a)$$

Forward pass:

$$z = wx + b, \qquad a = \sigma(z), \qquad L = L(a)$$

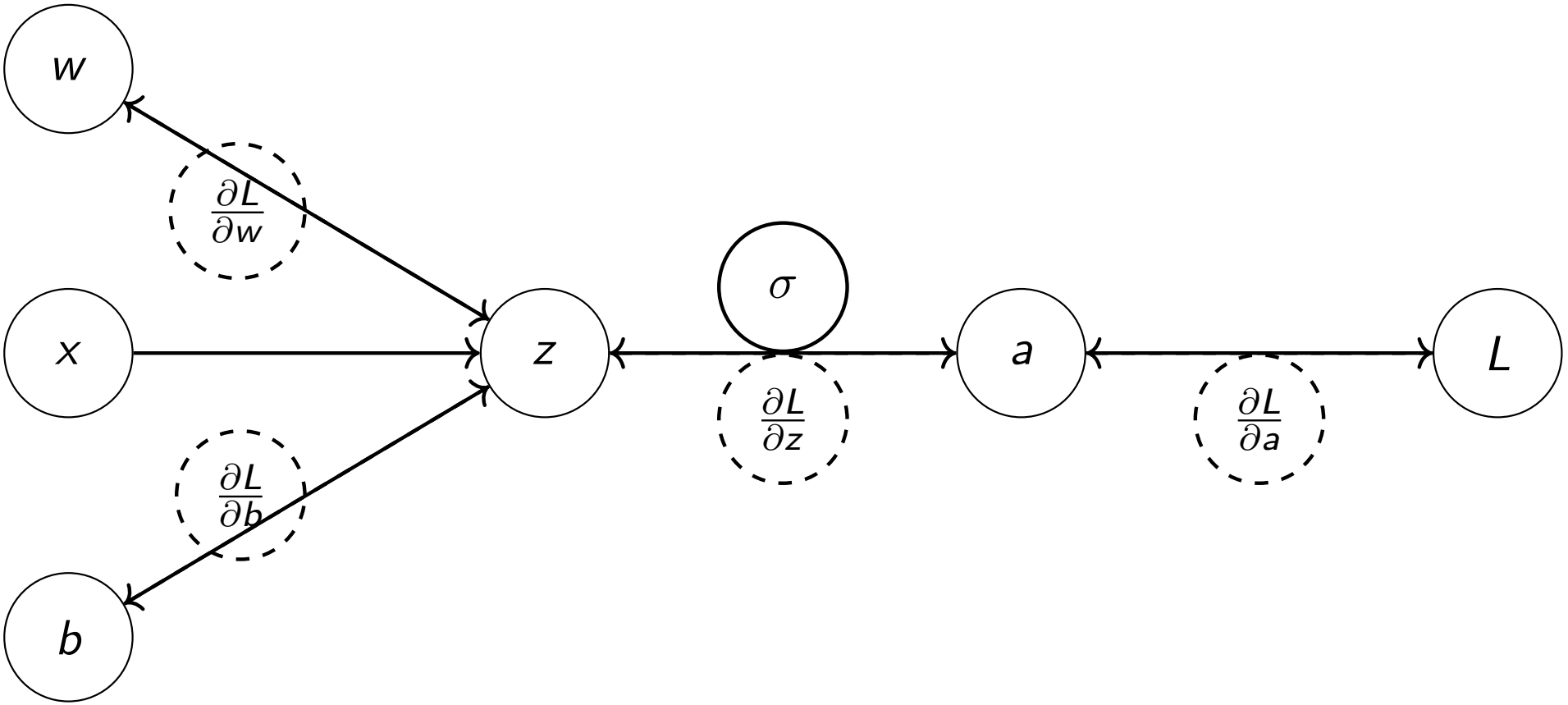
Backward Pass (Chain Rule):

$$w,\ b \longrightarrow z \longrightarrow a \longrightarrow L$$

Gradient:

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

Computation Graph



Gradient and Gradient Descent

For weights:

$$\boldsymbol{w} = [w_1, w_2, \ldots, w_n]$$

Gradient:

$$\nabla_{\boldsymbol{w}} \mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \frac{\partial \mathcal{L}}{\partial w_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_n} \end{bmatrix}$$

Direction of steepest loss increase.

Gradient Descent Update Rule

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}$$

- η – learning rate
- Move opposite to gradient

Why Taylor Series?

- Approximates complex functions using polynomials
- Enables local linear or quadratic analysis
- Forms the basis of optimization methods
- Widely used in deep learning theory

Key idea: Approximate a function near a point using derivatives.

Taylor Series: Definition

Let $f(x)$ be a function with derivatives of all orders at $x = a$. The Taylor series expansion of $f(x)$ about $x = a$ is:

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \cdots$$

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x - a)^n$$

Example: Taylor Series of $f(x) = e^x$

Derivatives of $f(x) = e^x$:

$$f'(x) = e^x, \quad f''(x) = e^x, \quad \ldots$$

At $x = 0$:

$$f^{(n)}(0) = 1 \quad \forall n$$

Taylor Expansion of e^x

Substitute into the Maclaurin series:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Polynomial approximation:

- First order: $1 + x$
- Second order: $1 + x + \frac{x^2}{2}$

First-Order Taylor Approximation

Linear approximation:

$$f(x) \approx f(a) + f'(a)(x - a)$$

Used in:

- Gradient descent
- Local sensitivity analysis

Second-Order Taylor Approximation

Quadratic approximation:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2$$

Used in:

- Newton's method
- Curvature analysis

Numerical Example: Problem Statement

Approximate the function

$$f(x) = \ln(1 + x)$$

around $x = 0$ using a Taylor series, and estimate the value at:

$$x = 0.1$$

Approach:

- Compute derivatives at $x = 0$
- Use first- and second-order Taylor approximations

Step 1: Compute Derivatives

Given:

$$f(x) = \ln(1 + x)$$

Derivatives:

$$f'(x) = \frac{1}{1+x}, \quad f''(x) = -\frac{1}{(1+x)^2}$$

At $x = 0$:

$$f(0) = 0, \quad f'(0) = 1, \quad f''(0) = -1$$

Step 2: First-Order Taylor Approximation

The first-order Taylor expansion about $x = 0$ is:

$$f(x) \approx f(0) + f'(0)x$$

Substitute values:

$$\ln(1 + x) \approx x$$

Numerical estimate at $x = 0.1$:

$$\ln(1.1) \approx 0.1$$

Step 3: Second-Order Taylor Approximation

The second-order Taylor expansion is:

$$f(x) \approx f(0) + f'(0)x + \frac{1}{2}f''(0)x^2$$

Substitute values:

$$\ln(1+x) \approx x - \frac{x^2}{2}$$

Numerical estimate at $x = 0.1$:

$$\ln(1.1) \approx 0.1 - \frac{(0.1)^2}{2} = 0.095$$

Step 4: True Value and Error

True value:

$$\ln(1.1) \approx 0.09531$$

- First-order approximation: 0.10000
- Second-order approximation: 0.09500

Observation:

- Second-order Taylor approximation is much more accurate
- Higher-order terms reduce approximation error

Deep Learning Interpretation

- First-order Taylor $\rightarrow$ Gradient Descent
- Second-order Taylor $\rightarrow$ Newton / curvature-aware methods
- Deep learning mostly relies on first-order methods due to efficiency

Key Insight:

$$\text{Optimization} = \text{Local Taylor Approximation of Loss Function}$$

Activation Functions

Function	Expression	Derivative
Sigmoid	$\sigma(x)$	$\sigma(x)(1-\sigma(x))$
ReLU	$\max(0,x)$	$1_{x>0}$
Tanh	$\tanh(x)$	$1-\tanh^2(x)$

Summary

- Derivatives measure sensitivity
- Chain rule enables backpropagation
- Gradients guide learning
- Gradient descent minimizes loss

Without calculus, deep learning cannot learn.