African Institute For Mathematical Sciences
Master in Machine Intelligence

Understanding Deep Neural Networks: Forward, Backward Pass and Residual Connections

Joshua Juste Emmanuel Yun Pei NIKIEMA

June 15, 2025

Abstract

Deep neural networks (DNNs) are powerful machine learning models inspired by the human brain. Despite their popularity, the mathematical principles behind them are often misunderstood or hidden behind libraries. In this short report, we present the architecture of deep neural networks, explain the forward and backward propagation mathematically, and introduce the concept of residual (skip) connections that help train very deep networks.

1 Deep Neural Network Architecture

A deep neural network (DNN) is composed of multiple layers of neurons. Each layer performs an affine transformation followed by a nonlinear activation.

Let the input vector be $x \in \mathbb{R}^d$, and consider a DNN with L layers. For each layer $l \in \{1, \ldots, L\}$, we define:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$
$$a^{(l)} = \sigma(z^{(l)})$$

where:

- $W^{(l)}$ is the weight matrix of layer l,
- $b^{(l)}$ is the bias vector,
- $a^{(l-1)}$ is the activation from the previous layer ($a^{(0)} = x$),
- σ is a nonlinear activation function (e.g., ReLU, sigmoid, tanh),
- $z^{(l)}$ is the pre-activation, and $a^{(l)}$ is the output of the layer.

Illustration: 3-Layer Network

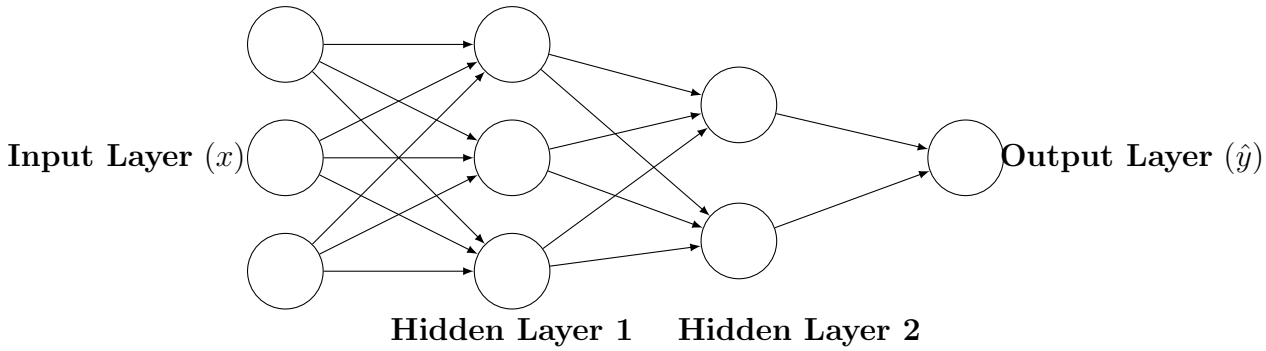

Figure 1: Feedforward Neural Network with 3 Layers: Each neuron is connected to all neurons in the next layer.

2 Forward Pass

In the forward pass, we compute the activations layer by layer from the input to the output.

Example with 3 layers:

$$\begin{aligned} z^{(1)} &= W^{(1)}x + b^{(1)} \\ a^{(1)} &= \sigma(z^{(1)}) \\ z^{(2)} &= W^{(2)}a^{(1)} + b^{(2)} \\ a^{(2)} &= \sigma(z^{(2)}) \\ \text{Output: } y_{\text{pred}} &= a^{(2)} \end{aligned}$$

3 Loss and Backward Pass

We define a loss function $\mathcal{L}(y, y_{\text{pred}})$ [1], for example, Mean Squared Error (MSE):

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

The backward pass uses the chain rule to compute gradients [2] and update weights using gradient descent:

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)}(a^{(l-1)})^T$$

where the error term $\delta^{(l)}$ is computed recursively:

$$\begin{aligned} \delta^{(L)} &= \nabla_{a^{(L)}}\mathcal{L} \circ \sigma'(z^{(L)}) \\ \delta^{(l)} &= \left((W^{(l+1)})^T\delta^{(l+1)}\right) \circ \sigma'(z^{(l)}) \end{aligned}$$

4 Residual (Skip) Connections

As networks get deeper, they suffer from vanishing gradients and degradation problems. **Residual connections** [3] help by allowing gradients to flow directly through layers.

Definition

Instead of learning a direct mapping $\mathcal{H}(x)$, the network learns the residual $\mathcal{F}(x) := \mathcal{H}(x) - x$, and outputs:

$$y = \mathcal{F}(x) + x$$

This simple change helps stabilize training for very deep networks.

Mathematical Insight

The gradient of the loss with respect to the input is now:

$$\frac{d\mathcal{L}}{dx} = \frac{d\mathcal{L}}{dy}\left(\frac{d\mathcal{F}(x)}{dx} + I\right)$$

This identity term I ensures that gradients do not vanish even when $\frac{d\mathcal{F}(x)}{dx}$ is small.

Illustration

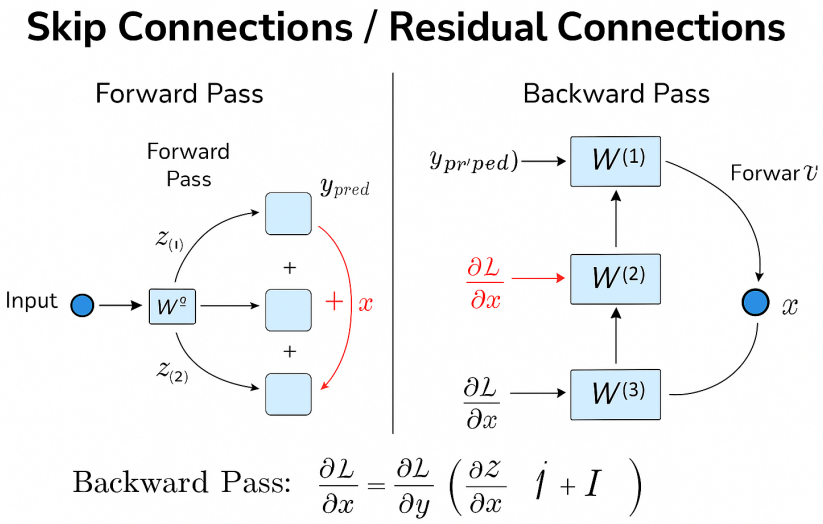

Figure 2: Illustration of the forward pass and backward pass in a neural network with residual (skip) connection.

Figure 2 provides a visual summary of the computational steps in a neural network that incorporates a *skip connection*.

- On the left, we observe the **forward pass**, where the input x goes through a linear transformation followed by a non-linear activation. The result is then *added* to the original input. This direct addition between the input and the transformed output is called a **residual connection**.
- On the right, the **backward pass** shows how gradients are propagated back through the network. Thanks to the shortcut from the input x, the gradient $\frac{\partial \mathcal{L}}{\partial x}$ receives an additional direct contribution, which helps preserve the gradient signal during training.

Mathematically, backpropagation through a residual connection can be expressed as:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y} \left(\frac{\partial z}{\partial x} \cdot I + I \right)$$

where I is the identity matrix, and z denotes the intermediate transformation without the residual connection. This formulation highlights how the direct path (via I) enhances gradient flow, thus mitigating the vanishing gradient problem.

5 Conclusion

We introduced the architecture of deep neural networks, derived the forward and backward pass mathematically, and presented the concept of residual connections. Understanding these principles helps demystify how deep learning models learn from data.

References

[1] From Maximum Likelihood to Loss Functions in Machine Learning by Joshua J. NIKIEMA. Available at: LinkedIn post

[2] Optimization in AI: The Mathematical Foundations of Gradient Descent and Momentum by Joshua J. NIKIEMA. Available at: LinkedIn post

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. Available at: https://www.deeplearningbook.org/