Nonlocal Approaches to Neural Network Training

Shaghayegh Rouhi
Faculty Advisor: Prof. Petronela Radu

Research Supported by UCARE

UNIVERSITY of NEBRASKA LINCOLN

Intro to Machine Learning

Machine Learning (ML) is a type of AI that allows computers to learn from data and make decisions without explicit programming.

Deep Learning is a branch of ML that uses multi-layered neural networks to model complex patterns and relationships in large datasets.

Neural Networks (NN): Inspired by the human brain, composed of layers of interconnected nodes (neurons).

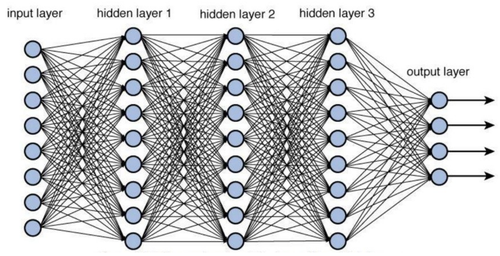


Figure 1. Neural Network Structure

Activation Functions add non-linearity to NNs, enabling them to learn complex patterns. Common examples: ReLU, sigmoid, and tanh.

Rectified Linear Unit (ReLU) is a popular activation function due to its simplicity and efficiency.

Standard ReLU and Its Limitations:

$$\text{ReLU}(Z) = \max(0, Z) \tag{1}$$

Its derivative is:

$$\text{ReLU}'(Z) = \begin{cases} 0, & Z < 0 \\ 1, & Z > 0 \\ \text{undefined}, & Z = 0 \end{cases} \tag{2}$$

The discontinuity at $Z = 0$ causes several issues:

- **Instability**: Backpropagation may lead to erratic weight updates.
- **Dead Neurons**: Neurons with negative inputs stop learning.
- **Sharp Gradient Transitions**: Sudden changes delay optimization.

Nonlocal Models - Motivations & Applications

- Allow derivatives of discontinuous functions.
- **Dynamic fracture**: Crack growth or structural changes
- **Image processing**: Computers analyze images using algorithms
- **Flocking and swarms**

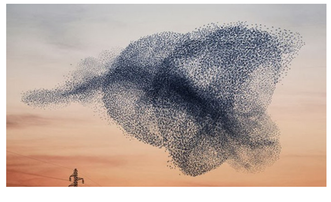


Figure 2. Swarming

Figure 3. Fragmented Glass

Mathematical Background

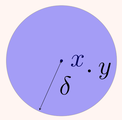
Nonlocal Derivative - For an interaction kernel μ that is odd define

$$D_\mu u(x) := \int_{-\delta}^{\delta} [u(x+z) - u(x)]\mu(z)\, dz,$$

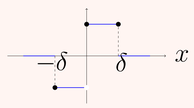
which approximates $u'(x)$ as $\delta \to 0$. (A. Haar, P. Radu. *A new nonlocal calculus framework*, 2022)

Examples of Kernels

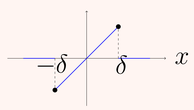
The point x interacts with neighbors y within δ through $\mu(y - x)$.



$$\mu_1(x) = \begin{cases} 0, & |x| > \delta \\ 1, & 0 < x < \delta \\ -1, & -\delta < x < 0 \end{cases}$$



$$\mu_2(x) = \begin{cases} x, & |x| < \delta \\ 0, & |x| > \delta \end{cases}$$



Example:

$$D_{\mu_1} ReLu(x) = \begin{cases} 0, & x < -\delta \\ x/\delta + 1/2 + x^2/2\delta^2, & -\delta < x < 0 \\ x/\delta + 1/2 - x^2/2\delta^2, & -\delta < x < 0 \\ 1, & x > \delta \end{cases}$$

Model Structure

Model Overview

- **Dataset**: MNIST
- **Input**: 784 features (28x28 image), normalized by 255.
- **Hidden Layer**: 10 neurons with ReLU activation.
- **Output Layer**: 10 neurons with Softmax activation.
- **Training**: Gradient descent, 1500 iterations, learning rate $\alpha = 0.1$.
- **Loss Function**: Cross-entropy with backpropagation.

ReLU Derivative Adjustment: We introduce a smooth, parameterized alternative using δ to approximate a continuous transition near zero.

Key Properties of the New ReLU Derivative

- **Continuity**: Smooth transition reduces instability at $Z = 0$.
- **Nonlocal Dependence**: The response incorporates neighboring values for better gradient flow.
- **Quadratic Approximation**: Improves training stability without high computational cost.

Effect of Smoothing Parameter δ

- **Small** $\delta \to$ Approximates standard ReLU (less smooth).
- **Larger** $\delta \to$ More smoothing, may slow training like tanh/sigmoid.

Experiments & Results - Nonlocal ReLu

```python
def ReLU_deriv(Z):
    return (Z > 0).astype(float)
```

Figure 4. Standard ReLU function

```python
def ReLU_deriv(Z, delta=0.1):
    return np.where(Z <= -delta, 0,  # Z < -delta, return 0
                    np.where(Z >= delta, 1,  # Z > delta, return 1 (m)
                             np.where(Z < 0, (Z / delta) - 1 / 2 + (Z**2 / 2 * delta**2),  # Z between -delta and 0 (*m)
                                      (Z / delta) - 1 / 2 + (Z**2 / 2 * delta**2))))  # Z between 0 and delta (*m)
```

Figure 5. Adjusted Nonlocal ReLU function

Accuracy	ReLU	d/dx ReLU
89%	Standard	Standard
92%	ReLU	D(ReLU)

Table 1. Accuracy Comparison of Standard vs. Nonlocal ReLU

Accuracy	α	Iteration
54%	0.01	1500
89.90%	0.05	1500
92%	0.1	1500
94.80%	**0.5**	**1500**
94.60%	0.7	1500
89%	1	1500

Table 2. Accuracy Results for Different Learning Rates (α) in the Nonlocal ReLU

Original			Nonlocal		
Type	Iteration	Time	Type	Iteration	Time
original	500	3m 31s	nonlocal	500	6m 32s
Accuracy	98.0%		Accuracy	97.7%	
Alpha	0.5		Alpha	0.5	
original	600	4m 4.7s	nonlocal	600	10m 11s
Accuracy	98.8%		Accuracy	98.5%	
Alpha	0.65		Alpha	0.65	

Table 3. Comparison of Original vs Nonlocal ReLU Function at Different Iterations

Conclusions & Future Work

Benefits of This Approach:

- Stabilizes training and avoids dead neurons
- Enables smooth, predictable gradient updates
- Higher computational accuracy by avoiding undefined points
- Simple to integrate into existing models

Applications:

- Deep and reinforcement learning
- Low-precision and scientific computing

Future Work:

- Test across kernels and activations
- Apply to more real-world ML tasks/data-sets