A Novel Theory of the Mechanism of Non-Linearity in Activation Functions

A Comprehensive Theory for Neural Networks

Draft Version: This paper is preliminary and has not yet been peer-reviewed.

James J Kalafus

August 28, 2024

While the non-linear nature of neural networks is widely recognized as being caused by non-linear activation functions, the precise mechanism by which these functions introduce non-linearity has not been explained until now. We propose a novel theory that the crucial **non-linearity of neural networks arises from activation functions disruption of the associativity of tensor operations**. This disruption of associativity, in concert with the manifold hypothesis, motivates researching new mathematical frameworks that can enhance neural network architecture.

We also leverage functional decompositions to analyze the vector space of activation functions. We determine that Gaussian/Dirac Delta and cosine functions serve as the fundamental modes for generating numerically stable activation functions via integration.

Associative Property

The concept that order of calculation doesn't change results:

$$\mathrm{A} \cdot (\mathrm{B} \cdot \mathrm{C}) = (\mathrm{A} \cdot \mathrm{B}) \cdot \mathrm{C} \quad \textbf{(1)}$$

A. **Core Concept — Disruption of Associativity:** *A non-linear activation function, applied element-wise, disrupts the associativity of tensor operations, preventing input variables from decoupling from weight parameters, making it* ***impossible to simplify sequential layers into a single equivalent transformation****. This irreducibility is the core mechanism by which non-linear activation functions instill the characteristic non-linearity of neural networks.*

$\mathrm{B} \cdot \mathrm{A}(x) = (\mathrm{B} \cdot \mathrm{A})(x)$ **no** activation function →multiple layers simplify **(2a)**

$\mathrm{B} \cdot \mathscr{A}(\mathrm{A} \cdot x) \neq (\mathrm{B} \cdot \mathscr{A} \circ \mathrm{A})(x)$ non-associative $\mathscr{A}$ →multiple layers **don't** simplify **(2b)**

linear ℓ →layers simplify

$$\text{input} \to \mathscr{T}_1 \to \ell \to \mathscr{T}_2 \to \text{output}$$

$$\downarrow \text{ can simplify} \quad \textbf{(2c)}$$

$$(\ell \circ \mathscr{T}_2)$$

non-linear $\mathscr{A}$ →layers **don't** simplify

$$\text{input} \to \mathscr{T}_1 \to \mathscr{A} \to \mathscr{T}_2 \to \text{output} \quad \textbf{(2d)}$$

$$\not\downarrow \text{ can}\mathbf{\text{not}} \text{ simplify}$$

$$(\mathscr{A} \circ \mathscr{T}_2)$$

The meaning of "non-linearity" in this paper is context dependent, and can refer to two closely intertwined but distinct concepts:

B. **Activation Function Non-Linearity:** *An activation function $\mathscr{A}(x)$ is considered non-linear in this context if its derivative $\frac{d}{dx}\mathscr{A}(x)$ is not equal to a constant k for all x in the range of $\mathscr{A}(x)$:* $$\frac{d}{dx}\mathscr{A}(x) \neq k \,\forall x \,\middle|\, k \in \text{Range}(\mathscr{A}) \quad \textbf{(3a)}$$

Contrariwise, some linear function $\ell(x)$ defined by its constant derivative

$$\frac{d}{dx}\ell(x) = k \,\forall x \,\middle|\, k \in \text{Range}(\ell) \quad \textbf{(3b)}$$

would not affect the associativity of any such tensor operations:

$\text{B} \cdot \ell(\text{A} \cdot x) = \big(\text{B} \cdot \ell \circ \text{A}\big)(x)$ linear function → multiple layers simplify **(3c)**

C. **Neural Network Non-Linearity:** *This refers to the phenomenon where, due to the introduction of non-linear activation functions, the tensor operations within each layer cannot be decoupled from the input variables. This prevents the simplification of multiple layers into a single equivalent layer. As a result, the neural network exhibits a higher-order transformation, where each layer's tensor transformation must be applied sequentially to the input, compounding linear effects into non-linear effects, ultimately resulting in what can be described as super-linear effects. In other words:*

$$\text{B} \cdot (\mathscr{A} \circ (\text{A} \cdot x)) \neq (\text{B} \cdot (\mathscr{A} \circ \text{A})) \cdot x \quad \textbf{(4)}$$

N.B., matrices are 2nd order tensors and vectors are 1st order tensors, and tensor contraction is matrix multiplication.

Proofs of the Disruption of Associativity

Formal Definition and Proof A

- To formally demonstrate how ***non-linear activation functions disrupt the associativity of tensor operations***, we focus first on tensor contraction. This proof will show that **the introduction of non-linear activation functions prevents the combination of sequential transformations into a single equivalent operation**, a key mechanism underlying the non-linearity in neural networks.

1. Sequential Linear Transformations

Let A and B be linear transformations (in this case, tensor contractions) acting on a tensor x. The composition of these transformations can be expressed as:

$$y \equiv \mathrm{B}\left(\mathrm{A}(x)\right) = \mathrm{B} \cdot (\mathrm{A} \cdot x) \quad \textbf{(5)}$$

Since both A and B are linear transformations, their composition is also a linear transformation because linear transformations are associative:

$$y = (\mathrm{B} \cdot \mathrm{A}) \cdot x \quad \textbf{(6)}$$

This shows that two sequential linear transformations can be expressed as a single linear transformation.

Suppose instead A_i are a recursive sequence of n linear transformations (in this case, tensor contractions) acting on a tensor x. Then, we can write:

$$y_i \equiv \mathrm{A}_i \cdot y_{i-1} \text{ where } y_0 \equiv \mathrm{A}_0 \cdot x \quad \textbf{(7)}$$

Since A_i are linear transformations, their composition is also a linear transformation, again because linear transformations are associative. First we show this intuitively, and then we prove the lemma inductively:

$$y_n = \mathrm{A}_n \cdot y_{n-1} = (\mathrm{A}_n \cdot \mathrm{A}_{n-1}) \cdot y_{n-2} = (\mathrm{A}_n \cdot \mathrm{A}_{n-1} \cdot \mathrm{A}_{n-2}) \cdot y_{n-3} = \ldots = \left(\prod_{i=0}^{n-1} \mathrm{A}_i\right) \cdot x \quad \textbf{(8)}$$

This expresses how the composition of any finite sequence of linear transformations can be expressed as a single linear transformation. Formally, we show by induction that this is true for base case $n = 1$:

$$y_0 = \left(\prod_{i=0}^{0} \mathrm{A}_i\right) \cdot x = \mathrm{A}_0 \cdot x \quad \textbf{(9)}$$

and that if it is true for n, then it must also be true for $n + 1$:

$$y_n = \left(\prod_{i=0}^{n-1} \mathrm{A}_i\right) \cdot x \implies y_{n+1} = \mathrm{A}_n \cdot y_n = \mathrm{A}_n \cdot \left(\prod_{i=0}^{n-1} \mathrm{A}_i\right) \cdot x = \left(\prod_{i=0}^{n} \mathrm{A}_i\right) \cdot x \quad \textbf{(10)}$$

2. Non-Linear Activation and Disruption of Tensor Algebra Associativity

Now, consider what happens when we interpose a non-linear element-wise activation function $\mathscr{A}$ between transformations A and B. The output would now be:

$$z \equiv \mathrm{B}\left(\mathscr{A}\left(\mathrm{A}(x)\right)\right) \quad \textbf{(11)}$$

Because $\mathscr{A}$ is non-linear, the expression $\mathscr{A}\left(\mathrm{A}(x)\right)$ does not associate in the same way linear functions do: $$z = \mathrm{B} \cdot (\mathscr{A} \circ (\mathrm{A} \cdot x)) \neq (\mathrm{B} \cdot (\mathscr{A} \circ \mathrm{A})) \cdot x \quad \textbf{(12)}$$

This inequality illustrates that you can no longer combine the sequential linear transformations A and B into a single equivalent transformation because the non-linear activation function $\mathscr{A}$ disrupts the associativity of the tensor operations.

Similarly, for a recursive sequence of activated transformations:

$$z_i \equiv \mathscr{A}_i\left(\mathrm{A}_i \cdot z_{i-1}\right) \text{ where } z_0 \equiv \mathscr{A}_0\left(\mathrm{A}_0 \cdot x\right) \quad \textbf{(13)}$$

Because $\mathscr{A}_i$ are non-linear, the expressions $\mathscr{A}_i\left(\mathrm{A}_i(z_{i-1})\right)$ do not associate in the same way compositions of linear functions do:

$$z_i = \mathscr{A}_i \circ \left(\mathrm{A}_i \cdot z_{i-1}\right) \neq \left(\mathscr{A}_i \circ \mathrm{A}_i\right) \cdot z_{i-1} \quad \textbf{(14a)}$$

$$z_n \neq \left(\prod_{i=0}^{n-1} \mathscr{A}_i \circ \mathscr{T}_i\right) \cdot x \quad \textbf{(14b)}$$

This inequality illustrates that you can no longer combine the linear transformations A_i into a single equivalent transformation because the non-linear activation functions $\mathscr{A}_i$ disrupt the associativity of the tensor operations.

Formal Definition and Abstract Proof B

- To generalize ***the disruption of associativity caused by non-linear activation functions***, we extend our analysis beyond tensor contraction to arbitrary linear tensor operations, including addition, Hadamard product, Kronecker product, and tensor product. This proof will demonstrate that **the introduction of a non-linear activation function prevents the combination of sequential transformations into a single equivalent operation**, that key mechanism underlying the non-linearity in neural networks.

1. General Linear Tensor Functions

Let $\mathscr{T}$ represent an arbitrary linear tensor operation, such as:

- tensor contraction (e.g. matrix multiplication)
- tensor product (a kind of outer product)
- addition (element-wise)
- Hadamard product (element-wise multiplication)
- Kronecker product (like outer product, but increases dimension)

2. Application of a Non-Linear Activation Function

Now, introduce a non-linear activation function $\mathscr{A}$ that is applied element-wise to the result of a linear tensor operation:

$$\mathbf{Z} = \mathscr{A}\left(\mathscr{T}(\mathbf{X})\right) \quad \textbf{(15)}$$

3. Disruption of Associativity in General Tensor Operations

To demonstrate that the disruption of associativity by $\mathscr{A}$ applies to any tensor operation $\mathscr{T}$, consider the following scenario. For any two tensor operations $\mathscr{T}_1$ and $\mathscr{T}_2$, and a tensor $\mathbf{X}$, the composition of these operations without the activation function is given by:

$$\mathscr{T}_2\left(\mathscr{T}_1(\mathbf{X})\right) = (\mathscr{T}_2 \circ \mathscr{T}_1)(\mathbf{X}) \quad \textbf{(16)}$$

Since $\mathscr{T}_1$ and $\mathscr{T}_2$ are linear, their composition $\mathscr{T}_2 \circ \mathscr{T}_1$ is also linear. However, when a non-linear activation function $\mathscr{A}$ is applied between these operations:

$$\mathbf{Z} = \mathscr{T}_2\left(\mathscr{A}\left(\mathscr{T}_1(\mathbf{X})\right)\right) \quad \textbf{(17)}$$

the associativity of $\mathscr{T}_1$ and $\mathscr{T}_2$ is disrupted. Specifically, this means:

$$\mathscr{T}_2\left(\mathscr{A} \circ \left(\mathscr{T}_1(\mathbf{X})\right)\right) \neq \left(\mathscr{T}_2\left(\mathscr{A} \circ \mathscr{T}_1\right)\right)(\mathbf{X}) \quad \textbf{(18)}$$

This inequality shows that the overall operation $\mathscr{T}_2\left(\mathscr{A}\left(\mathscr{T}_1(\mathbf{X})\right)\right)$ is not equivalent to any linear operation on the input tensor $\mathbf{X}$. The non-linear activation function $\mathscr{A}$ disrupts the ability to simplify the sequential operations $\mathscr{T}_1$ and $\mathscr{T}_2$ into a single operation.

4. Extending to All Tensor Operations

The key idea is that the non-linear function $\mathscr{A}$ disrupts associativity regardless of the specific nature of $\mathscr{T}$. Since the definition of linearity is consistent across all tensor operations (tensor contraction, tensor product, addition, Hadamard product, and

Kronecker product), the argument applies universally. This general form of associativity disruption is sufficient to cover all tensor functions that satisfy the linearity condition.

Proof C by Contradiction

- Demonstrate that **a non-linear activation function disrupts associativity, precluding reducing sequential tensor operations to a single equivalent operation**.

1. Assume the Contrary

Assume that a non-linear activation function $\mathscr{A}$ does not disrupt associativity and allows for the combination of arbitrary linear tensor operations into a single equivalent tensor operation. Specifically, for tensor operations $\mathscr{T}_1$ and $\mathscr{T}_2$ applied to a tensor $\mathbf{X}$:

$$\mathscr{T}_2\left(\mathscr{A} \circ \left(\mathscr{T}_1(\mathbf{X})\right)\right) = \left(\mathscr{T}_2\left(\mathscr{A} \circ \mathscr{T}_1\right)\right)(\mathbf{X}) \quad \textbf{(19)}$$

2. Express the Composition Without the Activation Function

Without the activation function $\mathscr{A}$, the composition of $\mathscr{T}_1$ and $\mathscr{T}_2$ would simply be:

$$\mathscr{T}_2\left(\mathscr{T}_1(\mathbf{X})\right) = \left(\mathscr{T}_2 \cdot \mathscr{T}_1\right)(\mathbf{X}) \quad \textbf{(20)}$$

This holds because tensor operations $\mathscr{T}_1$ and $\mathscr{T}_2$ are linear operations, and the composition of linear operations is associative.

3. Introduce the Activation Function

Now, we apply the non-linear activation function $\mathscr{A}$ between these operations:

$$\mathscr{T}_2\left(\mathscr{A} \circ \left(\mathscr{T}_1(\mathbf{X})\right)\right) \quad \textbf{(21)}$$

For this expression to be equivalent to the combined operation $\mathscr{T}_2\left(\mathscr{A} \circ \mathscr{T}_1\right)$, the activation function $\mathscr{A}$ must associate with the tensor operations, implying:

$$\mathscr{T}_2\left(\mathscr{A} \circ \left(\mathscr{T}_1(\mathbf{X})\right)\right) = \left(\mathscr{T}_2\left(\mathscr{A} \circ \mathscr{T}_1\right)\right)(\mathbf{X}) \quad \textbf{(22)}$$

4. Reach a Contradiction

However, by the definition of a non-linear function, $\mathscr{A}$ does not associate with $\mathscr{T}_1$ and $\mathscr{T}_2$. If it did, $\mathscr{A}$ would behave as a linear function, which contradicts the assumption that $\mathscr{A}$ is non-linear.

Thus, the original assumption that a non-linear activation function allows tensor operations to combine into a single equivalent operation must be false. This contradiction

demonstrates that non-linear activation functions inherently disrupt the ability to combine tensor operations into a single equivalent operation via the associative property. This affirms their role in introducing non-linearity by disrupting the associativity of tensor algebra in neural networks.

Connecting to Deeper Understanding

This novel theory that **activation functions disrupt the associative properties of tensor operations** in neural networks provides a deeper understanding of neural network non-linearity and opens new avenues for research. The foundational disruption is crucial for neural networks' ability to model complex, non-linear relationships, driving their success in tasks like image recognition, natural language processing, and reinforcement learning.

Implications for Neural Network Research

The manifold hypothesis, which posits that real-world data lies on a low-dimensional manifold within a higher-dimensional space, is reinvigorated by the insights from this novel theory. By recognizing that **neural networks operate as systems of non-associative transformations**, we can employ new mathematical frameworks to improve neural network design, allowing models to better recognize and utilize underlying linear structures in data embedded within complex, non-linear manifolds. The updated context of non-associative tensor algebra presents an opportunity to **rethink machine learning from a structural and relational perspective** inline with the manifold hypothesis. This approach holds promise for far more efficient training processes, improved generalization, and a deeper comprehension of how neural networks process information.

- *The Vector Space of Potential Non-Linear Activation Functions*

Let's consider for a moment the vector space of functions in general to explore potential activation functions. To explore other classes of functions, let's focus on Taylor series, Laurent series, and Fourier series; as this includes all infinitely differentiable

periodic and non-periodic functions, this analysis should adequately characterize the vector space.

Periodic Activation Functions?

Fourier Series represent functions as sums of sine and cosine terms:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos(n x) + b_n \sin(n x) \right) \quad \textbf{(23a)}$$

While Fourier series are not commonly recognized as activation functions, the SIREN activation function based on the odd $\left[f(-x) = -f(x)\right] \sin(x)$ function has demonstrated utility in image representation tasks. The perception that periodic functions might be "useless" for machine learning is challenged by such cases. This suggests periodic activation functions can work effectively in certain domains, though the even $\left[f(-x) = f(x)\right] \cos(x)$ has yet to find its applications. In fact, periodic activation functions represent an under-explored area that may warrant further research. Please see Sitzmann et.al. for the SIREN reference.

Non-Periodic Activation Functions

Laurent series include terms with negative powers, which introduce singularities:

$$f(x) = \sum_{n=-\infty}^{\infty} a_n (x - x_0)^n \quad \textbf{(23b)}$$

Singularities in Laurent series can cause computational problems due to the introduction of infinities. As a result, these series are generally unsuitable as activation functions. Singularities lead to numerical instability and are intractable in gradient-based methods.

Taylor series approximate non-periodic continuous functions using polynomials:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} (x - x_0)^n \quad \textbf{(23c)}$$

In practice, non-saturating functions only work as activation functions when they avoid causing the gradient to vanish or explode. Functions dominated by a polynomial term of degree two or higher have been found to cause the gradient to explode or vanish, destabilizing training.

It's important to note that while the theory suggests polynomial-based activations are feasible, in practice they cause gradient instability when higher-order terms dominate. Although an activation function must have non-constant derivatives **(3a)** in order to disrupt associativity, to maintain stable training it must simultaneously avoid polynomial contributions of degree two or higher. Among polynomials *with a finite number of terms* (why?), this is only achievable with piecewise linear functions.

Characterization of Piecewise Linear Functions for Numerical Stability

To ensure numerical stability in neural network training, it is crucial that activation functions and their derivatives do not introduce instability through gradient explosion or vanishing. *One* effective way to guarantee stability is by constraining activation functions to be bounded by piecewise linear functions.

Definition of Piecewise Linear Functions: A piecewise linear function is defined as a function composed of multiple linear segments, each defined over a specific interval of the input domain. Formally, a piecewise linear activation function $\mathscr{A}(x)$ can be expressed as:

$$\mathscr{A}(x) = \begin{cases} m_0 x + c_0, & x \in [a_0, b_0) \\ m_1 x + c_1, & x \in [a_1, b_1) \\ \vdots & \\ m_n x + c_n, & x \in [a_n, b_n) \end{cases} \quad \textbf{(24a)}$$

where m_i and c_i are constants that define the slope and intercept of each linear segment, and the intervals $[a_i, b_i)$ partition the domain of x.

Stability Boundaries: For an activation function $\mathscr{A}(x)$ and its derivative $\mathscr{A}'(x)$ to maintain stability, they must be bounded above and below by piecewise linear functions. This ensures that the gradient, which drives the learning process, remains within a predictable range, avoiding scenarios where gradients either explode or vanish.

A piecewise linear bounded activation function is formally defined by these constraints:

$$\begin{aligned} m_0 x + c_{0_{\min}} \le \mathscr{A}(x) \le m_0 x + c_{0_{\max}}, \quad x \in [a_0, b_0) \\ m_1 x + c_{1_{\min}} \le \mathscr{A}(x) \le m_1 x + c_{1_{\max}}, \quad x \in [a_1, b_1) \\ \vdots \\ m_n x + c_{n_{\min}} \le \mathscr{A}(x) \le m_n x + c_{n_{\max}}, \quad x \in [a_n, b_n) \end{aligned} \tag{24b}$$

$$\begin{aligned} m'_{0_{\min}} \le \mathscr{A}'(x) \le m'_{0_{\max}}, \quad x \in [a_0, b_0) \\ m'_{1_{\min}} \le \mathscr{A}'(x) \le m'_{1_{\max}}, \quad x \in [a_1, b_1) \\ \vdots \\ m'_{n_{\min}} \le \mathscr{A}'(x) \le m'_{n_{\max}}, \quad x \in [a_n, b_n) \end{aligned} \tag{24c}$$

where m_i, $c_{i_{\min}}$, $c_{i_{\max}}$, $m'_{i_{\min}}$, and $m'_{i_{\max}}$ are the slopes and intercepts of the bounding piecewise linear functions and their derivatives, respectively. These bounds guarantee:

A. Gradient Clipping: *By constraining $\mathscr{A}'(x)$ within linear bounds, the activation function can effectively "clip" extreme gradients, preventing runaway values that lead to gradient explosion.*

B. Mitigation of Vanishing Gradients: *Constraining $\mathscr{A}'(x)$ to be piecewise provides some degree of gradient flow, which mitigates the potential for a vanishing gradient problem that can stall training. By ensuring that the derivative $\mathscr{A}'(x)$ is bounded within a non-zero range across some intervals, we guarantee that the gradient does not approach zero universally.*

If a candidate activation function and its derivative are bounded above and below by piecewise linear functions, they are viable in the context of neural network training. This characterization lends flexibility to the design of custom activation functions that can then deviate piecewise from strict linearity while ensuring they remain within a numerically stable range.

Role of Gradient Stability in Activation Functions

It is well-established in the literature that linear functions provide stability in gradient propagation due to their constant rate of change. As detailed in works such as Goodfellow et al. (2016), Glorot & Bengio (2010), and Benign et al. (1994), linear functions

maintain a consistent gradient magnitude, avoiding the pitfalls of vanishing or exploding gradients that are common with higher-degree polynomials.

These principles underpin the effectiveness of piecewise linear functions as activation functions. By constraining both the activation function and its derivative to be bounded by piecewise linear segments, we ensure a degree of gradient flow balanced between gradient clipping, which could otherwise cause gradients to explode, and mitigating the risk of vanishing gradients, which could otherwise stall training.

In our novel framework, we build upon this established knowledge, showing that this constraint to be bounded by piecewise linear segments is not just a practical consideration but a fundamental aspect of how **non-linear activation functions disrupt the associative property of tensor operations**. Threading this needle ensures the neural network's non-linearity while preserving the stability necessary for deep learning.

- *Practical Non-Linear Activation Functions in Generality*

  Deriving the Fundamental Modes of Periodic Activation Functions

  Integrating $\int \cos(x)$ generates the $\sin(x)$ function.

  Deriving the Fundamental Modes of Piecewise Linear Activation Functions

We know that activation functions must be non-linear, but we also understand that if they are characteristically super-linear—meaning they cannot be bounded above and below by piecewise linear segments, including their derivatives—they lose numerical stability. Therefore, we need to explore the piecewise non-linear functions where all segments are linear. There are 3 types of piecewise non-linearities: singularities, non-singular discontinuities (steps; referred to as discontinuities for brevity), and corners (discontinuities of the derivative). We could start from any one of these and use calculus to derive the others; I chose to start with the Dirac Delta/Gaussian because they have the fewest parameters / degrees of freedom.

The Direct Delta function is of special interest because its first and second integrals generate the two relevant types of piecewise linear functions: discontinuities and corners. The **Dirac Delta function** δ is a piecewise singularity such that:

$$\delta(x - x_0) \equiv \begin{cases} 0, & x \neq x_0 \\ \infty, & x = x_0 \end{cases} \text{ where } \int_{-\infty}^{x} \delta(q - x_0) \cdot f(q) \cdot dq \equiv \begin{cases} 0, & x < x_0 \\ f(x_0), & x \geq x_0 \end{cases} \quad \textbf{(25a)}$$

Its definite integral generates **Heaviside** $\mathscr{H}$:

$$\mathscr{H}(x - x_0) \cdot f(x_0) \equiv \int_{-\infty}^{x} \delta(q - x_0) \cdot f(q) \cdot dq = \begin{cases} 0, & x < x_0 \\ f(x_0), & x \geq x_0 \end{cases} \quad \textbf{(25b)}$$

Then, integrating again, generates **ReLU**:

$$\mathrm{ReLU}(x - x_0) \cdot f(x_0) \equiv \int_{-\infty}^{x} \int_{-\infty}^{x} \delta(q - x_0) \cdot f(q) \cdot dq = \begin{cases} 0, & x < x_0 \\ f(x_0) \cdot x, & x \geq x_0 \end{cases} \quad \textbf{(25c)}$$

If we continue integration, we'll generate gradient destabilizing super-linear terms.

We can make linear combinations of these functions, yielding $\mathrm{Heaviside}$ recentered on zero, manifestly positive (or negative) $\mathrm{Heaviside}$, $\mathrm{LeakyReLU}$, and abs. N.B. that $\mathrm{Heaviside}$ all of the same sign and abs are not documented as activation functions.

As an ideal singularity, the Dirac Delta function is not an applicable activation function. The Dirac Delta function is defined in terms of the $\lim_{\sigma \to 0}$ of its smooth version, the **Gaussian function** $\mathscr{G}$:

$$\lim_{\sigma \to 0} \int_{-\infty}^{x} \mathscr{G}(q - x_0) \cdot f(q) \cdot dq = \int_{-\infty}^{x} \delta(q - x_0) \cdot f(q) \cdot dq \text{ where } \mathscr{G}(x) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad \textbf{(26)}$$

Therefore, we can generate equations with similar properties to those generated by integrating the Dirac Delta by instead integrating the Gaussian function, and recover them exactly in the $\lim_{\sigma \to 0}$. The Gaussian function integrates to the erf function, an S-curve contour in the sigmoid family that recovers the $\mathrm{Heaviside}$ step function in the $\lim_{\sigma \to 0}$:

$$\int_{-\infty}^{x} \mathscr{G}(x) \cdot dx = \frac{1}{2} \mathrm{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + \frac{1}{2} \quad \textbf{(27a)} \qquad \lim_{\sigma \to 0} \int_{-\infty}^{x} \mathscr{G}(x) \cdot dx = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \quad \textbf{(27b)}$$

Then, integrating again, we get a contour in the ReLU family, but with a Gaussian bump due to the smooth transition instead of the discontinuity:

$$\int_{-\infty}^{x}\int_{-\infty}^{x}\mathcal{G}(x)\cdot dx = \frac{x}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)+1\right)+\sqrt{2}\sigma^2\cdot\mathcal{G}(x) \quad \textbf{(28a)}$$

In the $\lim_{\sigma\to 0}$, we recover ReLU exactly:

$$\lim_{\sigma\to 0}\int_{-\infty}^{x}\int_{-\infty}^{x}\mathcal{G}(x)\cdot dx = \begin{cases}0, & x<0\\ x, & x\geq 0\end{cases} \quad \textbf{(28b)}$$

We couldn't take the derivative of the Dirac Delta, but we can take the derivative of the Gaussian. This makes for an odd $\left[f(-x)=-f(x)\right]$ variant of the even $\left[f(-x)=f(x)\right]$ Gaussian that constitutes a non-monotonic critical damping term. By tuning σ, it has potential to add a non-monotonic contour to the $\mathrm{sigmoid}$ and $\tanh$ family of S-curves.

$$\frac{d}{dx}\mathcal{G}(x) = \frac{-x}{\sigma^2}\mathcal{G}(x) \quad \textbf{(29)}$$

Consistency of the State of the Art with Derived Activation Functions

There are three basic families of non-periodic activation functions (Saturating Singularity, Saturating S-Curve, and Non-Saturating Piecewise Linear), and perhaps (experimental area) two fundamental families of periodic activation functions (even and odd). **Please see the Glossary for a comprehensive list of activation functions.**

Functions described thus far give a good survey. This derivation includes all the known shapes, and suggests that a $\mathrm{Heaviside}$ function of all of the same sign and the abs function could be useful activation functions. On the other hand, it may not be a coincidence that the only two candidate activation functions indicated by the Dirac Delta/Gaussian generation by integration method that have not been documented return values that are manifestly of one sign; this may be indicative that practical activation functions must be not only non-linear **(3)**, but also that they cannot be all of the same sign:

$$\exists\,(a,b),(c,d)\subset\mathrm{Domain}(f)\;\middle|\;\begin{pmatrix}\forall a'\in(a,b),\forall c'\in(c,d),\\ f(a')\cdot f(c')\leq 0 \text{ and}\\ f(a')\neq f(c')\end{pmatrix} \quad \textbf{(30)}$$

The Gaussian bump is an unwelcome feature in the Sigmoid-like S-curve, not quite fitting documented practice, but is an interesting feature in light of the landmark state of the art non-monotonic Swish and Mish activation functions. Non-monotonicity reportedly introduces greater flexibility in learning complex patterns, making Swish and Mish the non-saturating activation functions of choice for the most complex tasks such as vision and natural language processing (NLP). Swish aka SiLU was independently discovered by two lines of inquiry. Mish was first published claiming that it has advantages over Swish.

The non-monotonicity of the high-performance Swish and Mish functions introduces a previously unstudied critical damping contour. The idea of "critical damping" raises a key question: is this strategy as good as non-saturating activation gets? Could the Gaussian bump have foreshadowed non-monotonicity? We would be well-served to research critically damped saturating S-curves.

Advancing the State of the Art of Experimental Activation Functions

The ongoing search for effective activation functions focuses on maintaining a non-constant derivative without causing destabilizing growth. As recent discoveries suggest, there are likely more activation functions to be found. For example, Ramachandran, et. al., propose one method of Searching for Activation Functions. Non-monotonic sigmoid and tanh could prove effective and warrant further exploration.

Wavelet series could also serve as a basis for deriving activation functions that stabilize numerically sensitive regions of arbitrary non-periodic functions. This approach involves two aspects: selecting the characteristic function for multi-scale activation and determining which specific wavelets comprise the activation function.

While this paper provides several starting points, it intentionally leaves room for creative exploration. The less we constrain future research, the more we open the door to innovation. We leave you with two calls to action, depending on your skillset and inclination: one questing for experimental activation functions, and another as follows.

Deep Understanding & Broader Implications for Research

The novel theory that activation functions disrupt the associative properties of tensor operations offers a deeper understanding of non-linearity in neural networks. In the updated context of non-associative tensor algebra, **the manifold hypothesis guides us in reinterpreting machine learning from a structural and relational perspective.** The non-associativity introduced by non-linear activation functions opens the door to apply **new non-associative mathematical frameworks** previously considered unrelated to neural network design that we can now recognize as potentially applicable. This theory, therefore, establishes itself as a foundational concept with far-reaching implications, potentially shaping the next generation of machine learning architectures.

Glossary of Documented Activation Functions

- Saturating — Singularity Contours **instantaneous**
  - $\mathcal{G}(x) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$ range: $(0, \frac{1}{\sqrt{2\pi\sigma^2}})$ probability distributions
  - $\text{RBF}(x) \equiv e^{-\gamma|x-c|^2}$ range: $(0,1)$ similarity to reference c
  - $\text{sinc}(x) \equiv \frac{\sin(x)}{x}$ range: $(-0.217,1)$ experimental
  - $\frac{d}{dx}\mathcal{G}(x) = -\frac{x}{\sigma^2}\mathcal{G}(x)$ range: $(\frac{\pm 1}{\sigma\sqrt{2\pi\sigma^2 e}})$ experimental

- Saturating — S-Curve / Step Function Contours **binary**
  - $\text{sigmoid}(x) \equiv \sigma(x) \equiv \frac{1}{1+e^{-x}}$ range: $(0,1)$ binary classification
  - $\text{hard}\sigma(x) \equiv \begin{cases} 0, & x \le -0.5m \\ \frac{x+0.5m}{m}, & |x| < 0.5m \\ 1, & x \ge 0.5m \end{cases}$ range: $(0,1)$ $m \in (2.5,3.0)$ binary classification
  - $\tanh(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}}$ range: $(-1,1)$ zero-centered binary
  - $\text{hardtanh}(x) \equiv \begin{cases} -1, & x \le -1 \\ x, & |x| < 1 \\ 1, & x \ge 1 \end{cases}$ range: $(-1,1)$ zero-centered binary
  - $\text{softsign}(x) \equiv \frac{x}{1+|x|}$ range: $(-1,1)$ zero-centered binary
  - $\text{erf} \equiv \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ range: $(-1,1)$ experimental zero-centered binary
  - $\arctan(x)$ range: $(-\frac{\pi}{2}, \frac{\pi}{2})$ experimental zero-centered binary

- Non-Saturating — Piecewise Linear Contours **unbounded**
  - $\text{softplus}(x) \equiv \ln(1 + e^x)$ range: $(0,\infty)$ deep learning
  - $\text{swish}(x) \equiv x \cdot \sigma(x)$ range: $(-0.278,\infty)$ deep learning, vision, NLP reinforcement learning
  - $\text{hardswish}(x) \equiv x \cdot \text{hardSigmoid}(x)$ range: $(\frac{-m}{16}, \infty)$ deep learning, vision, NLP reinforcement learning
  - $\text{mish}(x) \equiv x \cdot \tanh\left(\ln(1 + e^x)\right)$ range: $(-0.308,\infty)$ deep learning, vision, NLP
  - $\text{ReLU}(x) \equiv \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$ range: $(0,\infty)$ deep learning, *sparse activation*
  - $\text{LeakyReLU}(x) \equiv \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$ range: $(-\infty, \infty)$ deep learning
  - $\text{ELU}(x) \equiv \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$ range: $(-\alpha, \infty)$ deep learning
  - $\text{abs}(x) \equiv |x|$ range: $(0,\infty)$ experimental

- Periodic Contours (largely experimental) **periodic**
  - $\sin(x)$ range: $(-1,1)$ continuous signals, vision
  - $\cos(x)$ range: $(-1,1)$ experimental
  - $\text{sinc}(x) \equiv \frac{\sin(x)}{x}$ range: $(-0.217,1)$ experimental

Bibliography

Foundational Concepts

Non-Linearity and Activation Functions in Depth;
Numerical Stability & Polynomials;
Manifold Hypothesis

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/

Fourier, Laurent, and Taylor Function Decompositions;
Non-Associativity of Non-Linear Tensor Operations

- Riley, K. F., Hobson, M. P., & Bence, S. J. (2006). *Mathematical Methods for Physics and Engineering: A Comprehensive Guide* (3rd ed.). Cambridge University Press.

Numerical Stability & Polynomials

Numerical Stability & Polynomials

- Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166. doi:10.1109/72.279181

Numerical Stability & Polynomials

- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), 9, 249-256. https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

Manifold Hypothesis

Various methods for testing the manifold hypothesis and its implications for machine learning.

- Fefferman, C., Mitter, S. K., & Narayanan, H. (2016). Testing the Manifold Hypothesis. Journal of the American Mathematical Society, 29(4), 983-1049. https://www.ams.org/journals/jams/2016-29-04/S0894-0347-2016-00845-1/

Relates the manifold hypothesis to the structure of learned representations in deep networks.

- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798-1828. https://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf

Activation Functions

HardTanh

- Bishop, C. M. (1995). Neural Networks for Pattern Recognition. Oxford University Press.

RBF, Gaussian

- Broomhead, D. S., & Lowe, D. (1988). Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. *Royal Signals and Radar Establishment (RSRE)*.

ELU

- Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv preprint arXiv:1511.07289*.

Softplus

- Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2001). Incorporating Second-Order Functional Knowledge for Better Option Pricing. *Advances in Neural Information Processing Systems (NeurIPS)*, 13.

Swish

- Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107, 3-11.

Hard Swish, Hard Sigmoid

- Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., … & Le, Q. V. (2019). Searching for MobileNetV3. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* https://openaccess.thecvf.com/content_ICCV_2019/html/Howard_Searching_for_MobileNetV3_ICCV_2019_paper.html

Tanh

- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

Softsign

- Glorot, X., & Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Leaky ReLU

- Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the 30th International Conference on Machine Learning (ICML)*.

Mish

- Misra, D. (2019). Mish: A Self Regularized Non-Monotonic Activation Function. *arXiv preprint arXiv:1908.08681*. https://arxiv.org/1908.08681

ReLU

- Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)*.

Swish, systematic search

- Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for Activation Functions. *arXiv preprint arXiv:1710.05941*. https://arxiv.org/1710.05941

Sigmoid

- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323, 533-536.

Sine

- Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., & Wetzstein, G. (2020). Implicit Neural Representations with Periodic Activation Functions. *arXiv preprint arXiv:2006.09661*. https://arxiv.org/abs/2006.09661