What Makes Neural Networks Trainable? Invexity as a Structural Design Principle in AI (Supplemental material)

Samuel Pinilla[1,†], Ana M. Sanabria[2], Jia Bi[1], and Karen Egiazarian[3]
[1]Rutherford Appleton Laboratory, Harwell Campus, Didcot OX11 0QX, UK
[2]Universidade de São Paulo, São Paulo, Brazil
[3]Tampere University, Tampere, Finland
email: [†]samuel.pinilla@stfc.ac.uk

Contents

A Activation Functions in AI

A.1 Convex Activation Functions

Convex activation functions, characterized by their mathematical property that any line segment between two points on the function lies above the curve, have played a pivotal role in shaping modern neural networks (see Table 1). Functions like the Rectified Linear Unit (ReLU) and its variants (e.g., Leaky ReLU), which are convex across their domains, are foundational to deep learning architectures due to their computational efficiency and favorable optimization dynamics. Their piecewise linear nature, particularly in ReLU, promotes sparsity by deactivating negative inputs, which streamlines computation and mitigates the vanishing gradient problem. While neural networks remain inherently non-convex systems, the adoption of convex activations has empirically balanced expressivity and trainability, contributing to breakthroughs in computer vision, natural language processing, and beyond.

Table 1: List of convex activation functions used in the AI literature.

Name	**Activation function**	**Range**	**Type**
TRec_t [1]	$\sigma_t(x) = \begin{cases} x & \text{if } x > t \\ 0 & \text{otherwise} \end{cases}, t > 0$	$[0, \infty)$	Convex
PReLU [2]	$\sigma(x) = \max\{\alpha x, x\}$, $\alpha > 0$	$(-\infty, \infty)$	

A.2 Invexity and Quasi-invexity Theory: List of Activation Functions

In this section, we introduce a systematic mathematical framework for characterizing activation functions that are *invex* and *quasi-invex*. To this end, we present several concepts needed for the theoretical developments in this paper, starting with the definition of a locally Lipschitz continuous function.

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz continuous at a point $\boldsymbol{x} \in \mathbb{R}^n$, if there exist scalars $K > 0$ and $\epsilon > 0$ such that for all $\boldsymbol{y}, \boldsymbol{z} \in B(\boldsymbol{x}, \epsilon)$ we have $|f(\boldsymbol{y}) - f(\boldsymbol{z})| \leq K\|\boldsymbol{y} - \boldsymbol{z}\|_2$.*

Since the ordinary directional derivative, the most important tool in optimization, does not necessarily exist for locally Lipschitz continuous functions, it is required to introduce the concept of subdifferential [3], which is calculated in practice as follows.

Theorem 1. *[3, Theorem 3.9] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function at $\boldsymbol{x} \in \mathbb{R}^n$, and define $\Omega_f = \{\boldsymbol{x} \in \mathbb{R}^n | f$ is not differentiable at the point $\boldsymbol{x}\}$. Then the subdifferential of f is given by*

$$\partial f(\boldsymbol{x}) = \ conv\ (\{\boldsymbol{\zeta} \in \mathbb{R}^n |\ exists\ (\boldsymbol{x}_i) \in \mathbb{R}^n \setminus \Omega_f\ such\ that\ \boldsymbol{x}_i \to \boldsymbol{x}\ and\ \nabla f(\boldsymbol{x}_i) \to \boldsymbol{\zeta}\})\,. \tag{1}$$

We note that the subdifferential is always nonempty for locally Lipschitz continuous functions [3, Theorem 3.3]. Based on these, the concept of an invex function is presented as follows.

Definition 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz; then f is invex if there exists a function $\eta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\forall \boldsymbol{\zeta} \in \partial f(\boldsymbol{y})$, we have $f(\boldsymbol{x}) - f(\boldsymbol{y}) \geq \boldsymbol{\zeta}^T \eta(\boldsymbol{x}, \boldsymbol{y})$.*

It is well known that a convex function simply satisfies this definition with $\eta(\mathbf{x}, \mathbf{y}) = \mathbf{x} - \mathbf{y}$.

Further generalization of invexity is possible; indeed, [4] introduced also the following class of generalized convex functions.

Definition 3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz; then f is said to be quasi-invex if there exists a function $\eta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ such that $f(\boldsymbol{x}) - f(\boldsymbol{y}) \leq 0 \implies \boldsymbol{\zeta}^T \eta(\boldsymbol{x}, \boldsymbol{y}) \leq 0$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\forall \boldsymbol{\zeta} \in \partial f(\boldsymbol{y})$.*

A final remark, that follows easily from the above Definition 3, is that any invex function is quasi-invex [4].

A.2.1 Identifying Invex Activation Functions

Here we provide the mathematical tools needed to identify the invex activation functions reported in Table 2.

Theorem 2. *Let $\sigma : \mathbb{R}^n \to \mathbb{R}$ be a function. If σ satisfies one of the following conditions*

1. *every stationary point of σ is a global minimizer [5, Theorem 2.2].*
2. *σ does not have stationary points [5, Theorem 2.2].*

3. for $n = 1$ if first derivative of σ satisfies $\sigma' > 0$ [6, Theorem 3].

then σ is invex.

It is critical to emphasize that the class of functions satisfying the third characterization in Theorem 2 exhibits closure under key algebraic operations –including summation and composition– thereby enabling a systematic methodology for constructing invex activation functions. This property marks a significant advancement, as such structured constructions were previously confined to the narrower domain of convex mappings. This result not only bridges a theoretical gap in generalized convexity but also unlocks practical opportunities for designing neural networks with rigorously interpretable optimization landscapes using non-convex activation functions.

Examples of invex activation functions: To streamline our exposition and avoid redundant technical repetition, we now demonstrate invexity for some activation function from Table 2. These illustrative examples elucidate how the tools developed in Theorem 2 systematically generalize to the broader class of functions enumerated in the table. Crucially, the remaining functions can be verified through analogous reasoning.

- **Softmax:** This function is defined as $\sigma_i : \mathbb{R}^n \to \mathbb{R}$ where $\sigma_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}[i]}}{\sum_{j=1}^n e^{\boldsymbol{x}[j]}}$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Here we prove $\sigma_i(\boldsymbol{x})$ is invex by showing that $\sigma_i(\boldsymbol{x})$ does not have stationary points. Specifically, notice that

$$\nabla\sigma_i(\boldsymbol{x}) = \left[\frac{-e^{\boldsymbol{x}[i]}e^{\boldsymbol{x}[1]}}{\left(\sum_{j=1}^n e^{\boldsymbol{x}[j]}\right)^2}, \cdots, \sigma_i(\boldsymbol{x})(1-\sigma_i(\boldsymbol{x})), \cdots, \frac{-e^{\boldsymbol{x}[i]}e^{\boldsymbol{x}[n]}}{\left(\sum_{j=1}^n e^{\boldsymbol{x}[j]}\right)^2}\right]^T. \tag{2}$$

  From the above equation it is clear that $\nabla\sigma_i(\boldsymbol{x}) \neq \mathbf{0}$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Thus, by Theorem 2 Softmax activation function is invex.

- **MaxOut:** This function is defined as $\sigma : \mathbb{R}^n \to \mathbb{R}$ where $\sigma(\boldsymbol{x}) = \max\{\boldsymbol{x}[i]\}$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Here we prove $\sigma(\boldsymbol{x})$ is invex by showing that $\sigma(\boldsymbol{x})$ does not have stationary points. Specifically, taking $r = \arg\max\{\boldsymbol{x}[i]\}$, notice that

$$\nabla\sigma(\boldsymbol{x}) = [0, \cdots, 1, \cdots, 0]^T, \tag{3}$$

  where the value 1 in the above equation is located in the rth-entry of $\boldsymbol{x}$. Thus, it is clear that $\nabla\sigma(\boldsymbol{x}) \neq \mathbf{0}$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Thus, by Theorem 2 the MaxOut activation function is invex.

- **Generalized Sigmoid:** This function is defined as $\sigma(x) = \frac{\alpha}{1+e^{-\beta x}}$ for $\alpha, \beta > 0$. Observe that $\sigma'(x) = \frac{\alpha\beta}{(1+e^{-\beta x})^2}$, and as a consequence $\sigma'(x) > 0$ since $\alpha, \beta > 0$. Thus, by Theorem 2 the Sigmoid activation function is invex.

- **Softplus:** This function is defined as $\sigma(x) = \frac{1}{\beta}\log(1+e^{\beta x})$. Observe that $\sigma'(x) = \frac{e^{\beta x}}{1+e^{\beta x}}$, and as a consequence $\sigma'(x) > 0$. Thus, by Theorem 2 the Softplus activation function is invex.

- **SiLU:** This function is defined as $\sigma(x) = \frac{x}{1+e^{-x}}$. Observe that $\sigma'(x) = \frac{e^x(x+e^x+1)}{(e^x+1)^2}$, and as a consequence solving $x + e^x + 1 = 0$ it gives $x \approx -1.27$. Additionally, $\sigma(-1.27) \approx -0.27$ which is the global minimum of $\sigma(x)$. Thus, by Theorem 2 the SiLU activation function is invex since the unique stationary point is the global minimizer.

- **Log-ReLU:** This function is defined as $\sigma(x) = \log(\beta\max\{x, 0\} + 1)$ for $\beta > 0$. Observe that $\sigma'(x) = 0$ for $x \leq 0$. Additionally, $\sigma(x)$ achieves his global minimum for every $x \leq 0$. Thus, by Theorem 2 the Log-ReLU activation function is invex since the every stationary point is a global minimizer.

Table 2: List of invex activation functions used in the AI literature

Name	**Activation function**	**Range**	**Type**
LogLog [7]	$\sigma(x) = e^{-e^{-x}}$	$(0, 1)$	
cLogLog [7]	$\sigma(x) = 1 - e^{-e^{-x}}$	$(0, 1)$	
Sigmoid Gumbe [8]	$\sigma(x) = \frac{e^{-e^{-x}}}{1+e^{-x}}$	$(0, 1)$	Invex
Soft clipping [9]	$\sigma(x) = \frac{1}{a}\log\left(\frac{1+e^{ax}}{1+e^{a(x-1)}}\right)$, $a > 0$	$(0, 1)$	
Rootsig [10]	$\sigma(x) = \frac{x}{1+\sqrt{1+x^2}}$	$(-1, 1)$	
Mish [11]	$\sigma(x) = x\tanh(\ln(1 + e^x))$	$[\approx -0.3, \infty)$	

Name	Activation function	Range	Type
LogSigmoid [12]	$\sigma(x) = \log\left(\frac{1}{1+e^{-x}}\right)$	$(-\infty, 0)$	Invex
Sigmoidal selector [13]	$\sigma_k(x) = \left(\frac{1}{1+e^{-x}}\right)^k$	$(0, 1)$	
Flatten-T Swish [14]	$\sigma(x) = \begin{cases} \frac{x}{1+e^{-x}} + T & \text{if } x > 0 \\ T & \text{otherwise} \end{cases}$	$[T, \infty)$	
LogSoftmax [15]	$\sigma_i(\boldsymbol{x}) = \log\left(\frac{e^{\boldsymbol{x}[i]}}{\sum_{j=1}^{n} e^{\boldsymbol{x}[j]}}\right)$	$(-\infty, 0)$	
Sigmoid-Algebraic [16]	$\sigma(x) = \frac{1}{1+e^{-\frac{x(1+a\lvert x\rvert)}{1+\lvert x\rvert(1+a\lvert x\rvert)}}}$, $a > 0$	$(0, 1)$	
arctan [17]	$\sigma(x) = \tan^{-1}(x)$	$(-\frac{\pi}{2}, \frac{\pi}{2})$	
Soft Exponential [18]	$\sigma(x) = \lambda \begin{cases} \frac{\log(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{-\alpha x}-1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$(-\infty, \infty)$	
Bent Identity [19]	$\sigma(x) = \frac{\sqrt{x^2+1}-1}{2} + x$	$(-\infty, \infty)$	
Inverse Square root [20]	$\sigma(x) = \begin{cases} x & \text{if } x \geq 0 \\ \frac{x}{\sqrt{1+\alpha x^2}} & \text{if } x < 0 \end{cases}$	$\left(-\frac{1}{\sqrt{\alpha}}, \infty\right)$	
Log-ReLU [21]	$\sigma(x) = \log(\beta \max\{x, 0\} + 1)$, $\beta > 0$	$[0, \infty)$	
MaxOut [22]	$\sigma(\boldsymbol{x}) = \max\{\boldsymbol{x}[i]\}$	$(-\infty, \infty)$	
Rootsig [10]	$\sigma(x) = \frac{x}{1+\sqrt{1+x^2}}$	$(-1, 1)$	
Suish [1]	$\sigma(x) = \max\{x, xe^{-\lvert x\rvert}\}$	$[\approx -0.36, \infty)$	
LiSHT [23]	$\sigma(x) = x\tanh(x)$	$[0, \infty)$	
Logish [24]	$\sigma(x) = x\log\left(1 + \frac{1}{1+e^{-x}}\right)$	$[\approx -0.25, \infty)$	
Gish [25]	$\sigma(x) = x\log(2 - e^{-e^x})$	$[\approx -0.27, \infty)$	
LogLogish [26]	$\sigma(x) = x(1 - e^{-e^x})$	$[\approx -0.31, \infty)$	
Self arctan [27]	$\sigma(x) = x\tan^{-1}(x)$	$[0, \infty)$	
SquarePlus [28]	$\sigma(x) = \frac{1}{2}\left(x + \sqrt{x^2+\epsilon}\right)$, $\epsilon \geq 0$	$(0, \infty)$	
ELU [29]	$\sigma(x) = \begin{cases} x & \text{if } x \geq 0 \\ e^x - 1 & \text{if } x < 0 \end{cases}$	$(-1, \infty)$	

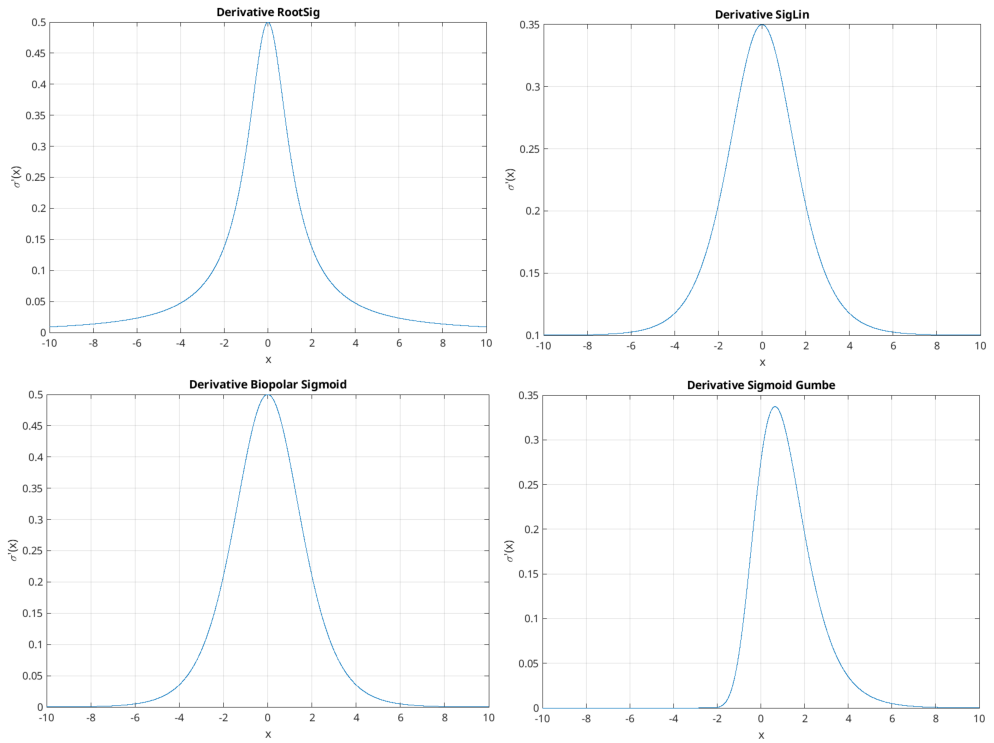


Figure 1: Here we present graphical proof of invexity for the RootSig, SigLin, Bipolar Sigmoid, and Sigmoid Gumbel activation functions, as illustrated in Figure 1. By leveraging third characterization of invex functions in Theorem 2, this figure shows that the derivative of these activation functions are strictly positive.

We conclude this section by providing a graphical proof of invexity for the RootSig, SigLin, Bipolar Sigmoid, and Sigmoid Gumbel activation functions, as illustrated in Figure 1. By leveraging the third characterization of invex functions in Theorem 2, Figure 1 illustrates that the derivative of these activation functions is strictly positive, thereby establishing their invexity. This visual proof not only complements our analytical framework but also underscores the versatility of invexity as a property that persists across diverse non-convex regimes.

A.2.2 Identifying Quasi-Invex Activation Functions

Here we provide a new mathematical tool needed to identify the quasi-invex activation functions reported in Table 3 summarized in the following theorem.

Theorem 3. *Let $f : \mathcal{D} \to \mathbb{R}$ be a locally Lipschitz function with $\mathcal{D} \subseteq \mathbb{R}^n$. If there exits an open set $\mathcal{X} \subseteq \mathcal{D}$ such that $\hat{f} : \mathcal{X} \to \mathbb{R}$, defined as $\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$, is invex, then f is quasi-invex.*

Proof. By hypothesis we know there exists an open set $\mathcal{X} \subseteq \mathcal{D}$ such that $\hat{f} : \mathcal{X} \to \mathbb{R}$, defined as $\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$, is invex. This means that there exists a function $\hat{\eta} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^n$ such that $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$, $\forall \boldsymbol{\zeta} \in \partial \hat{f}(\mathbf{y})$, we have

$$\hat{f}(\mathbf{x}) - \hat{f}(\mathbf{y}) \geq \boldsymbol{\zeta}^T \hat{\eta}(\mathbf{x}, \mathbf{y}). \tag{4}$$

Considering the above inequality we define $\eta : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^n$ as

$$\eta(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \hat{\eta}(\boldsymbol{x}, \boldsymbol{y}) & \text{if } \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \\ \mathbf{0} & \text{otherwise} \end{cases}. \tag{5}$$

As a consequence of (5) it is clear that

$$f(\boldsymbol{x}) - f(\boldsymbol{y}) \leq 0 \implies \boldsymbol{\zeta}^T \eta(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \tag{6}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{D}$, and all $\boldsymbol{\zeta} \in \partial f(\boldsymbol{y})$. Thus, the result holds. □

Examples of quasi-invex activation functions: To streamline our exposition and avoid redundant technical repetition, we now demonstrate quasi-invexity for some activation function from Table 3. These illustrative examples elucidate how the tool developed in Theorem 3 systematically generalize to the broader class of functions enumerated in the table. Crucially, the remaining functions can be verified through analogous reasoning.

- **Cosine:** Define $\sigma(x) = \cos(x)$. Take $\mathcal{X} = \left(\bigcup_{n\in\mathbb{N}}(2n\pi, 2(n+1)\pi)\right) \bigcup \left(\bigcup_{n\in\mathbb{N}}(-2(n+1)\pi, -2n\pi)\right)$. Observe that $\sigma(x)$ has only global minimizers in the set $\mathcal{X}$. Therefore, $\sigma(x)$ is invex in $\mathcal{X}$. Thus, appealing to Theorem 3 the cosine activation is quasi-invex.
- **Hard tanh:** Define $\sigma(x) = \max\{-1, \min\{1, x\}\}$. Take $\mathcal{X} = (-\infty, 1)$. Observe that $\sigma(x)$ has only global minimizers in the set $\mathcal{X}$. Therefore, $\sigma(x)$ is invex in $\mathcal{X}$. Thus, appealing to Theorem 3 the Hard tanh activation is quasi-invex.

Table 3: List of quasi-invex activation functions used in the AI literature.

Name	**Activation function**	**Range**	**Type**
Step [30]	$\sigma(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$	$\{0, 1\}$	Quasi-invex
Bipolar [31]	$\sigma(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$	$\{-1, 1\}$	

B Invexity Theory for Neural Networks

This section establishes a novel theoretical framework based on invexity analysis for characterizing the trainability of neural networks. We start by introducing the following preliminary results.

B.1 Preliminaries

We introduce the following definitions.

Definition 4. *A negative combination of $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r \in \mathbb{R}^n$ is a linear combination $\sum_{i=1}^r \lambda_i \boldsymbol{a}_i$ with $\lambda_i \leq 0$; if for all $\lambda_i < 0$ and all $\boldsymbol{a}_i \neq \boldsymbol{0}$ we say it is a strictly negative combination.*

Definition 5. *A set of vectors $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_r\} \subset \mathbb{R}^n$ is negatively dependent if at least one of the $\boldsymbol{a}_i$ is a negative combination of the others. Otherwise the set is negatively independent.*

Observe that negative independence is not an empty definition. In particular, any basis $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n\}$ of $\mathbb{R}^n$ is negatively independent. More significantly, the following theorem demonstrates that $\mathbb{R}^n$ admits negatively independent sets of cardinality exceeding the dimension n, establishing a fundamental difference from linear independence.

Theorem 4. *For any $n \in \mathbb{N}$ there exists at least $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{n+2} \in \mathbb{R}^n$ negatively independent vectors.*

Proof. We proceed by construction. Take a basis $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\} \subset \mathbb{R}^n$ that spans the space $\mathbb{R}^n$. Since $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ is linearly independent, then it is also negatively independent and $\boldsymbol{a}_i \neq \boldsymbol{0}$ for all $i = 1, \ldots, n$. Define $\boldsymbol{a}_{n+1} = \lambda_1 \boldsymbol{a}_1 + \cdots + \lambda_{n-1}\boldsymbol{a}_{n-1} - \lambda_n \boldsymbol{a}_n$, and $\boldsymbol{a}_{n+2} = \beta_1 \boldsymbol{a}_1 + \cdots - \beta_{n-1}\boldsymbol{a}_{n-1} + \beta_n \boldsymbol{a}_n$ where $\lambda_i, \beta_i > 0$ for all $i = 1, \ldots, n$. Then, it is clear that the set of vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{n+2}\}$ is negatively independent. Thus the result holds. □

The final results of this section, Theorems 5 and 6 establish foundational criteria for neural network invexity. These results furnish necessary conditions to characterize invex architectures, enabling the comprehensive framework derived in Theorem 7 (presented in the next section).

Theorem 5. *Let $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_r\} \subset \mathbb{R}^n$ be a set of negatively independent vectors where all $\boldsymbol{a}_i \neq \boldsymbol{0}$. Then, when $\lambda_i \geq 0, \forall i = 1, \ldots, r$ we get $\sum_{i=1}^r \lambda_i \boldsymbol{a}_i \neq \boldsymbol{0}$.*

Proof. We proceed by contradiction. Assume there exist $\lambda_i \geq 0$ for $i = 1, \ldots, r$ such that $\sum_{i=1}^r \lambda_i \boldsymbol{a}_i = \boldsymbol{0}$. Since, all vectors $\boldsymbol{a}_i \neq \boldsymbol{0}$ then at least there exits $j \in \{1, \ldots, r\}$ such that $\lambda_j > 0$. Then we get that

$$\begin{aligned} \lambda_j \boldsymbol{a}_j &= \sum_{i=1, i\neq j}^{r} -\lambda_i \boldsymbol{a}_i \\ \boldsymbol{a}_j &= \sum_{i=1, i\neq j}^{r} -\frac{\lambda_i}{\lambda_j} \boldsymbol{a}_i. \end{aligned} \tag{7}$$

The above equation implies that the set of vectors $\{\boldsymbol{a}_1, \cdots, \boldsymbol{a}_r\}$ is negatively dependent which is a contradiction. Then, when $\lambda_i \geq 0, \forall i = 1, \ldots, r$ we get $\sum_{i=1}^r \lambda_i \boldsymbol{a}_i \neq \boldsymbol{0}$. Thus the result holds. □

Theorem 6 (*Gale's theorem for linear inequalities* [32])**.** *For a given $p \times n$ matrix $\boldsymbol{A}$ and given vector $\boldsymbol{c} \in \mathbb{R}^p$ either*

1. *$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c}$ has a solution $\boldsymbol{x} \in \mathbb{R}^n$ or*
2. *$\boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{0}$, $\boldsymbol{c}^T\boldsymbol{\lambda} = -1$, has a solution $\boldsymbol{\lambda} \in \mathbb{R}^p_+$,*

but never both.

B.2 Invexity in Neural Networks

A neural network model is formally characterized as a parametric function $\mathcal{N}_\theta : \mathcal{X} \to \mathcal{Y}$, mapping an input $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ to an output $\boldsymbol{y} \in \mathcal{Y} \subset \mathbb{R}^m$, where θ denotes the learned fixed weights. The network's output is given by

$$\mathcal{N}_\theta(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]^T = \boldsymbol{y}, \tag{8}$$

where each component function $f_i : \mathcal{X} \to \mathcal{Z} \subset \mathbb{R}$ generates a scalar output, the Cartesian product $\mathcal{Z}^m \subseteq \mathcal{Y}$ defines the output space. Notice that, we avoid the use of notation θ within the functions $\{f_i(\boldsymbol{x})\}_{i=1}^m$ because they are fixed weights.

From model in (8), *we are interested in investigating the invexity of $\mathcal{N}_\theta$ with respect to its input $\boldsymbol{x} \in \mathcal{X}$, assuming fixed weights θ. The critical step lies in demonstrating that all component functions $\{f_i(\boldsymbol{x})\}_{i=1}^m$ are invex with respect to the same $\eta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ (see Definition 2).* To formalize this and demonstrate its feasibility, we introduce the proposition below, which establishes sufficient conditions for invexity in neural networks.

Theorem 7. *Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be differentiable functions for $i = 1, \ldots, r$ where their gradient satisfy $\nabla f_i(\boldsymbol{x}) \neq \boldsymbol{0}$ for all $\boldsymbol{x} \in \mathbb{R}^n$. If $\nabla f_1(\boldsymbol{x}), \ldots, \nabla f_r(\boldsymbol{x})$ are negatively independent for all $\boldsymbol{x} \in \mathbb{R}^n$ then functions $f_1, \ldots, f_r$ are invex with respect to the same $\eta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$.*

Proof. We proceed by contradiction. Assume no common function $\eta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ exists. Then, there exists $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ such that the system

$$\begin{aligned} f_1(\boldsymbol{x}) - f_1(\boldsymbol{y}) &\geq \eta(\boldsymbol{x}, \boldsymbol{y})^T \nabla f_1(\boldsymbol{y}) \\ &\vdots \\ f_r(\boldsymbol{x}) - f_r(\boldsymbol{y}) &\geq \eta(\boldsymbol{x}, \boldsymbol{y})^T \nabla f_r(\boldsymbol{y}), \end{aligned} \tag{9}$$

has no solution $\eta(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^n$. Rewrite the above inequality system as $\boldsymbol{A}\eta(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{c}$ where

$$\boldsymbol{A} = \begin{pmatrix} \nabla f_1(\boldsymbol{y})^T \\ \vdots \\ \nabla f_r(\boldsymbol{y})^T \end{pmatrix}, \text{ and } \boldsymbol{c} = \begin{pmatrix} f_1(\boldsymbol{x}) - f_1(\boldsymbol{y}) \\ \vdots \\ f_r(\boldsymbol{x}) - f_r(\boldsymbol{y}) \end{pmatrix},$$

with $\boldsymbol{A} \in \mathbb{R}^{r \times n}$, and $\boldsymbol{c} \in \mathbb{R}^r$. By Gale's theorem of the alternative for linear inequalities in Theorem 6 there exists $\boldsymbol{\lambda} \in \mathbb{R}^r_+$ such that $\boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{0}$. However, appealing to Theorem 5 we know that concluding $\boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{0}$ is a contradiction since by hypothesis the gradients $\{\nabla f_1(\boldsymbol{x}), \ldots, \nabla f_r(\boldsymbol{x})\}$ are negatively independent. Thus, we conclude that common $\eta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ exists such that all functions $f_1, \ldots, f_r$ are invex with respect to the same η. Therefore the result holds. □

Three critical insights emerge from Theorem 7: i) This result provides a general framework for analyzing neural network structures through invex optimization theory. Notably, it avoids restrictive assumptions common in convex network analysis (e.g., positivity constraints on weights or limitations on activation functions [33]). ii) The differentiability requirement for all $\{f_i^\theta(\boldsymbol{x})\}_{i=1}^m$ aligns with standard training paradigms (e.g., gradient descent, Adam [34]), underscoring the practical viability of our framework. iii) The negative independence of gradients $\{\nabla f_i^\theta(\boldsymbol{x})\}_{i=1}^m$ is a non-restrictive condition, as we demonstrate in a subsequent section by analyzing the Multilayer Perceptron architecture.

Considering that negative independence can be achieved by linear independence, the following is a natural consequence from Theorem 7.

Corollary 1. *Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be differentiable functions for $i = 1, \ldots, r$. If $\nabla f_1(\boldsymbol{x}), \ldots, \nabla f_r(\boldsymbol{x})$ are linearly independent for all $\boldsymbol{x} \in \mathbb{R}^n$ then functions $f_1, \ldots, f_r$ are invex with respect to the same $\eta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$.*

Proof. To prove this corollary it is only needed to observe that a linearly independent set is also negatively independent and non-zero set. Thus, the result holds. □

Lastly, we present the following which shows that Invexity is preserved through composition for any pair of differentiable functions when the rows of their Jacobians are linearly independent. This result is key to show that layer-wise invexity enables modular design of trainable architectures without sacrificing flexibility (i.e. this result will be employed in Sections E, and F).

Theorem 8. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$, and $g : \mathbb{R}^m \to \mathbb{R}^p$ be differentiable functions defined as*

$$f(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix}, \quad g(\boldsymbol{z}) = \begin{bmatrix} g_1(\boldsymbol{z}) \\ \vdots \\ g_p(\boldsymbol{z}) \end{bmatrix}, \tag{10}$$

for $f_i : \mathbb{R}^n \to \mathbb{R}$, and $g_j : \mathbb{R}^m \to \mathbb{R}$, with $p \leq m \leq n$, and $i = 1, \ldots, m$, $j = 1, \ldots, p$. If the Jacobian matrices $J_f(\boldsymbol{x}) \in \mathbb{R}^{m \times n}$, and $J_g(\boldsymbol{z}) \in \mathbb{R}^{p \times m}$ of functions f, and g respectively are full-row-rank for all $\boldsymbol{x} \in \mathbb{R}^n$, and $\boldsymbol{z} \in \mathbb{R}^m$ then functions

$$h_j(\boldsymbol{x}) = (g_j \circ f)(\boldsymbol{x}) = g_j(f(\boldsymbol{x})), \tag{11}$$

for $j = 1, \ldots, p$ are invex with respect to the same η.

Proof. Define function $h : \mathbb{R}^n \to \mathbb{R}^p$ as

$$h(\boldsymbol{x}) = \begin{bmatrix} h_1(\boldsymbol{x}) \\ \vdots \\ h_p(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} g_1(f(\boldsymbol{x})) \\ \vdots \\ g_p(f(\boldsymbol{x})) \end{bmatrix}. \tag{12}$$

Observe that the Jacobian matrix $J_h(\boldsymbol{x}) \in \mathbb{R}^{p\times n}$ of function $h(\boldsymbol{x})$ is given as

$$J_h(\boldsymbol{x}) = J_g(f(\boldsymbol{x}))J_f(\boldsymbol{x}) = \begin{bmatrix} \nabla h_1(\boldsymbol{x})^T \\ \vdots \\ \nabla h_p(\boldsymbol{x})^T \end{bmatrix}. \tag{13}$$

From the hypothesis $J_f(\boldsymbol{x})$, and $J_g(f(\boldsymbol{x}))$ are full-row-rank then vectors $\nabla h_1(\boldsymbol{x}), \ldots, \nabla h_p(\boldsymbol{x})$ are linearly independent for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore, by Corollary 1 functions $h_1(\boldsymbol{x}), \ldots, h_p(\boldsymbol{x})$ are invex with respect to the same η. Thus the result holds. □

B.3 Invexity for Multilayer Perceptron Building Block

The perceptron layer of a network $\mathcal{N}_\theta$ is defined as $\ell : \mathbb{R}^r \to \mathbb{R}^s$

$$\ell(\boldsymbol{z}) = \sigma\left(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}\right), \tag{14}$$

where $\boldsymbol{z} \in \mathbb{R}^r$ is the input, $\sigma : \mathbb{R} \to \mathbb{R}$ is a differentiable activation function applied element-wise, $\boldsymbol{W} \in \mathbb{R}^{r\times s}$ denotes the learnable weight matrix, and $\boldsymbol{b} \in \mathbb{R}^s$ the bias vector. We remark that the above model is valid when $\boldsymbol{W}$ denotes either a dense or a convolution layer. Considering the model in (14), we study its invexity in the following result.

Theorem 9. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function defined as*

$$f(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \sigma(\boldsymbol{w}_1^T\boldsymbol{x} + b_1) \\ \vdots \\ \sigma(\boldsymbol{w}_m^T\boldsymbol{x} + b_m) \end{bmatrix}, \tag{15}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a differentiable activation function, $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m]^T \in \mathbb{R}^{m\times n}$ (learning weights), $\boldsymbol{b} = [b_1, \ldots, b_m]^T \in \mathbb{R}^m$ (biases), and $m \leq n$. If σ does not have stationary points, and the rows of $\boldsymbol{W}$ are linearly independent then all $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are invex with respect to the same η.

Proof. Observe that the Jacobian matrix $J_f(\boldsymbol{x}) \in \mathbb{R}^{m\times n}$ of function f is given as

$$J_f(\boldsymbol{x}) = \operatorname{diag}([\sigma'(\boldsymbol{w}_1^T\boldsymbol{x} + b_1), \ldots, \sigma'(\boldsymbol{w}_m^T\boldsymbol{x} + b_m)])\boldsymbol{W}, \tag{16}$$

where the notation $\operatorname{diag}(\boldsymbol{v})$ builds a diagonal matrix where its entries are the vector $\boldsymbol{v}$. From hypothesis we know that $m \leq n$, and $\boldsymbol{W}$ is full-row-rank (i.e. all rows are linearly independent). Additionally, since σ does not have stationary points, that is $\sigma' \neq 0$, then by Corollary 1 we get that $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are invex with respect to the same η. Thus, the result holds. □

The last result presented in this section is a natural consequence of Theorem 9. Specifically, the assumption of σ not having stationary points can be easily achieved when σ is strictly increasing as summarized in the following corollary.

Corollary 2. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function defined as*

$$f(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \sigma(\boldsymbol{w}_1^T\boldsymbol{x} + b_1) \\ \vdots \\ \sigma(\boldsymbol{w}_m^T\boldsymbol{x} + b_m) \end{bmatrix}, \tag{17}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a differentiable activation function, $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m]^T \in \mathbb{R}^{m\times n}$ (learning weights), $\boldsymbol{b} = [b_1, \ldots, b_m]^T \in \mathbb{R}^m$ (biases), and $m \leq n$. If σ is strictly increasing, and the rows of $\boldsymbol{W}$ are linearly independent then all $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are invex with respect to the same η.

Proof. From hypothesis we know that σ is strictly increasing, then it means that $\sigma' > 0$ (i.e. does not have stationary points), therefore appealing to Theorem 9 we get that $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})$ are invex with respect to the same η. Thus, the result holds. □

B.4 Operations Preserving Invexity

In this section we present several operations that preserve invexity that are useful to prove trainability for neural networks.

Lemma 1. *Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ be a function defined as*

$$f(\boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} f_1(\boldsymbol{x}, \boldsymbol{z}) \\ \vdots \\ f_m(\boldsymbol{x}, \boldsymbol{z}) \end{bmatrix} = \begin{bmatrix} \sigma(\boldsymbol{w}_1^T \boldsymbol{x} + b_1) \\ \vdots \\ \sigma(\boldsymbol{w}_m^T \boldsymbol{x} + b_m) \end{bmatrix} + \boldsymbol{z}, \tag{18}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a differentiable activation function applied element-wise, $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m]^T \in \mathbb{R}^{m \times n}$ denotes the learned weight matrix, and $\boldsymbol{b} \in \mathbb{R}^m$ the bias vector. Then, $f_1(\boldsymbol{x}, \boldsymbol{z}), \ldots, f_m(\boldsymbol{x}, \boldsymbol{z})$ are invex with respect to the same η.

Proof. Observe that the Jacobian matrix $J_f(\boldsymbol{x}, \boldsymbol{z}) \in \mathbb{R}^m \times \mathbb{R}^{n+m}$ of function f is given as

$$J_f(\boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} \text{diag}([\sigma'(\boldsymbol{w}_1^T \boldsymbol{x} + b_1), \ldots, \sigma'(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)])\boldsymbol{W} & \boldsymbol{I} \end{bmatrix}. \tag{19}$$

Notice that $J_f(\boldsymbol{x}, \boldsymbol{z})$ as defined above is full-row-rank (because identity matrix is concatenated horizontally), which means that all rows of $J_f(\boldsymbol{x}, \boldsymbol{z})$ are linearly independent leading, by Corollary 1, to the invexity of $f_1(\boldsymbol{x}, \boldsymbol{z}), \ldots, f_m(\boldsymbol{x}, \boldsymbol{z})$ with respect to the same η. □

The above analytical result is key in order to prove Universal Approximation of Invex Neural Networks, in the next section.

Lemma 2. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$, and $g : \mathbb{R}^m \to \mathbb{R}^s$ be vector functions where $s < m \leq n$. Assume that the Jacobian matrix of g denoted as $\mathcal{J}_g(\boldsymbol{z}) \in \mathbb{R}^{s \times m}$ is full-row-rank for all $\boldsymbol{z} \in \mathbb{R}^m$, and that the row-rank r_f of the Jacobian matrix of f denoted as $\mathcal{J}_f(\boldsymbol{x}) \in \mathbb{R}^{m \times n}$ satisfies that $s \leq r_f < m$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Then, all component functions of the composed vector function $h(\boldsymbol{x}) = (g \circ f)(\boldsymbol{x})$ are invex with respect to the same η.*

Proof. Observe that the Jacobian matrix of function h denoted as $J_h(\boldsymbol{x}) \in \mathbb{R}^s \times \mathbb{R}^n$ is given as

$$J_h(\boldsymbol{x}) = J_g(f(\boldsymbol{x}))J_f(\boldsymbol{x}) = \begin{bmatrix} \nabla h_1(\boldsymbol{x})^T \\ \vdots \\ \nabla h_s(\boldsymbol{x})^T \end{bmatrix}. \tag{20}$$

From the hypothesis we can conclude that the row-rank of the Jacobian matrix $J_h(\boldsymbol{x})$ is s for all $\boldsymbol{x} \in \mathbb{R}^n$, which means that the vectors $\nabla h_1(\boldsymbol{x}), \ldots, \nabla h_s(\boldsymbol{x})$ are linearly independent leading, by Corollary 1, to the invexity of all component functions of the function $h(\boldsymbol{x}) = (g \circ f)(\boldsymbol{x})$ are invex with respect to the same η.

□

B.5 Universal Approximation of Invex Neural Networks

In this section we show that invex neural networks can universally approximate continuous functions which is summarized in the following result.

Theorem 10. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a nonaffine differentiable function that does not have stationary points (i.e. $\sigma' \neq 0$). Let $\mathcal{K} \subseteq \mathbb{R}^n$ be compact. Then, the class of invex neural networks $\mathcal{N}_\theta$ described by feedforward neural networks with n neurons in the input layer, m neurons in the output layer, and an arbitrary number of hidden layers, each with $n+m+2$ neurons with activation function σ is dense in $\mathcal{C}(\mathcal{K}; \mathbb{R}^m)$ with respect to the uniform norm.*

Proof. From hypothesis we have that σ is an invex differentiable function (see Theorem 2), therefore continuous, which means it satisfies the conditions of Theorem 3.2 from [35]. This means that the class of neural networks $\mathcal{N}_\theta$ described by feedforward neural networks with n neurons in the input layer, m neurons in the output layer, and an arbitrary number of hidden layers, each with $n + m + 2$ neurons with activation function σ is dense in $\mathcal{C}(\mathcal{K}; \mathbb{R}^m)$. Therefore, we need to show that this class of neural networks is invex.

To illustrate this, we recall that the hidden layers for this type of feedforward neural network are described, according to Theorem 3.2 from [35], as the consecutive composition of an affine mapping, σ, and an affine function. Mathematically, without loss of generality, the hidden layers for this type of feedforward neural network are given by

$$\ell(\boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} \sigma(\boldsymbol{w}_1^T \boldsymbol{x} + b_1) \\ \vdots \\ \sigma(\boldsymbol{w}_{m+n+2}^T \boldsymbol{x} + b_{m+n+2}) \end{bmatrix} + \boldsymbol{z}, \tag{21}$$

$\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n+m+2}]^T$ denotes the learned weight matrix, $\boldsymbol{b}$ the bias vector, and $\boldsymbol{z}$ belongs to the external affine function. Appealing to Lemma 1, we have that all component functions of all hidden layers $\ell(\boldsymbol{x}, \boldsymbol{z})$ are invex (i.e., Jacobian of $\ell(\boldsymbol{x}, \boldsymbol{z})$ is full-row-rank) with respect to the same η. Thus, by invoking Theorem 8 we conclude that all component functions of neural network $\mathcal{N}_\theta$ are invex with respect to the same η. Thus the result holds. □

The importance of the above analytical result is that by extending the universal approximation theorem to invex neural spaces, we demonstrate that invex networks retain full expressivity and can approximate any continuous function –without sacrificing architectural depth or flexibility.

C Proof of Invexity for ResNet

In this section, we provide proof of invexity for the ResNet structure. From Section 2.4 of the manuscript, we know that a ResNet structure $\mathcal{N}_\theta : \mathbb{R}^n \to \mathbb{R}^m$ is given by

$$\mathcal{N}_\theta(\boldsymbol{x}) = \left(\ell_5 \circ \underbrace{\ell_4 \circ \cdots \circ \ell_4}_{L_4} \circ \underbrace{\ell_3 \circ \cdots \circ \ell_3}_{L_3} \circ \underbrace{\ell_2 \circ \cdots \circ \ell_2}_{L_2} \circ \underbrace{\ell_1 \circ \cdots \circ \ell_1}_{L_1} \circ \ell \right)(\boldsymbol{x}), \tag{22}$$

where $L = S \cdot (L_1 + \cdots + L_4) + 2$ is the number of layers in ResNet, the last number two, in L, comes from the first 7×7 convolutional layer, and the fully connected layer at the end of the ResNet structure. The number S defines the weight layers in each ResNet block and determines the type of ResNet. Particularly, standard configurations (e.g., ResNet-50 with $S = 3$, $(L_1, L_2, L_3, L_4) = (3, 4, 6, 3)$) is an instance of this schema [36].

In the following analytical result, we prove the invexity of the ResNet structure.

Theorem 11. *Assume $\mathcal{N}_\theta$ is any ResNet structure as illustrated in Figure 2 of the manuscript, and formally modeled in Equation* (22)*. Then, $\mathcal{N}_\theta$ is invex.*

Proof. Concretely, we have to show that all functions $\ell, \ell_1, \ldots, \ell_4$, and ℓ_5 are differentiable and the rows of their Jacobian are linearly independent. To that end, we proceed on a case-by-case basis.

1. The first aspect to mention is that ResNet belongs to under-complete convolutional architectures [37] i.e. the number of filters do not exceed the dimension of the input. This implies that linearly independence within the convolutional layers is ensured.

2. Observe that function $\ell : \mathbb{R}^n \to \mathbb{R}^r$ with $r < n/2$ defined as $\ell(\boldsymbol{x}) = (g \circ f)(\boldsymbol{x})$. Here $f(\boldsymbol{x}) = \text{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where ReLU is applied element-wise, represents a convolutional transformation followed by a ReLU activation, where $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ denotes the learned convolutional filters and $\boldsymbol{b} \in \mathbb{R}^m$ the bias vector. The operator $g(\boldsymbol{z}) = \mathcal{M}(\boldsymbol{z})$ implements max pooling, with $g : \mathbb{R}^m \to \mathbb{R}^r$ reducing spatial dimensions.

   Following the methodology of Theorem 9, the Jacobian of f contains a block-diagonal structure contingent on the activation pattern of ReLU (i.e., inducing a binary diagonal matrix). Under batch normalization (which symmetrizes the distribution of $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$), half the entries of $\boldsymbol{h}$ are negative in expectation. Thus, rank of Jacobian of f, $\mathcal{J}_f(\boldsymbol{x})$, is $m/2$ in expectation for all $\boldsymbol{x}$.

   The max pooling operator g is a subsampling selector. Therefore, the Jacobian of g, $\mathcal{J}_g(\boldsymbol{z})$ is a binary matrix whose rows are linearly independent by construction.

   From the above analysis, we have that the functions f and g satisfy the assumptions of Lemma 2, which leads to the conclusion that the Jacobian of $\ell(\boldsymbol{x})$ is full-row-rank (i.e., all rows are linearly independent).

3. Observe that functions ℓ_1, ℓ_2, ℓ_3, and ℓ_4 in ResNet structure follow the residual ResNet block architecture. This means that $\ell_i(\boldsymbol{z}) = f_i(\boldsymbol{z}) + \boldsymbol{z}$ for $i = 1, \ldots, 4$, where $f_i(\boldsymbol{z})$ models the interaction between the weight layers and ReLU of ResNet block (see Figure 2(b) of the manuscript). This means that the Jacobian of $f_i(\boldsymbol{z}) + \boldsymbol{z}$ is given as $\boldsymbol{I} + \mathcal{D}f_i(\boldsymbol{z})$ where $\mathcal{D}f_i(\boldsymbol{z})$ is the Jacobian of $f_i(\boldsymbol{z})$, and $\boldsymbol{I}$ is the identity matrix. Since $f_i(\boldsymbol{z})$ entirely depends on the learned weights having complete freedom to choose its values, it is clear that $\boldsymbol{I} + \mathcal{D}f_i(\boldsymbol{z})$ is invertible, leading to a full-row-rank Jacobian matrix for all $\ell_i(\boldsymbol{z})$ for any $\boldsymbol{z}$.

4. Lastly, the function ℓ_5 is given the consecutive composition of an Average Pooling operation and a Dense layer. Formally, ℓ_5 is mathematically modeled as $\ell_5 = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{p \times s}$ contains both the Average Pooling operation and the Dense Layer with $s < p$, and $\boldsymbol{b} \in \mathbb{R}^s$ is the bias vector. Since $\boldsymbol{W}$ is the learnt weights and they have complete freedom to choose its values, it is clear that the Jacobian of ℓ_5, $\boldsymbol{W}$, is full-row-rank for any $\boldsymbol{x}$.

Considering the above analysis, we can invoke Theorem 8 which ensures that all component functions of any ResNet structure $\mathcal{N}_\theta$ are invex with respect to the same η. Thus, the result holds. □

As a final remark, it is clear from the above proof that ResNet without a skip connection and with ReLU activation function lacks an invex structure. Because when batch normalization precedes ReLU, it reduces the linear independence for rows (i.e., row-rank) of Jacobian by approximately half of a ResNet block without skip connection, that is, ReLU's nullification of negative values introduces stationary points that degrade the row-rank of Jacobian $\mathcal{D}\mathcal{H}(\boldsymbol{z})$.

D Proof of Invexity for UNet

In this section, we provide proof of invexity for the UNet structure. From Section 2.4 of the manuscript, we know that a UNet structure $\mathcal{N}_\theta : \mathbb{R}^n \to \mathbb{R}^m$ is given by

$$\mathcal{N}_\theta(\boldsymbol{x}) = \left(\ell_{12} \circ \tilde{\ell}_{11} \circ \cdots \circ \tilde{\ell}_9 \circ \tilde{\ell}_8 \circ \ell_7 \circ \cdots \circ \ell_2 \circ \ell_1 \circ \ell\right)(\boldsymbol{x}), \tag{23}$$

where the functions $\{\tilde{\ell}_i\}_{i=8}^{11}$ can be expressed as

$$\begin{aligned}
\tilde{\ell}_8(\boldsymbol{z}_7) &= \text{Concat}\left[\boldsymbol{z}_7, \ell_8(\ell_7(\boldsymbol{z}_7))\right], \\
\tilde{\ell}_9(\boldsymbol{z}_5) &= \text{Concat}\left[\boldsymbol{z}_5, \ell_9(\tilde{\ell}_8(\ell_6(\ell_5(\boldsymbol{z}_5))))\right], \\
\tilde{\ell}_{10}(\boldsymbol{z}_3) &= \text{Concat}\left[\boldsymbol{z}_3, \ell_{10}(\tilde{\ell}_9(\ell_4(\ell_3(\boldsymbol{z}_3))))\right], \\
\tilde{\ell}_{11}(\boldsymbol{z}_1) &= \text{Concat}\left[\boldsymbol{z}_1, \ell_{11}(\tilde{\ell}_{10}(\ell_2(\ell_1(\boldsymbol{z}_1))))\right].
\end{aligned} \tag{24}$$

To prove the invexity of the UNet structure it is clear that if all functions $\ell_1 \circ \ell, \ell_3 \circ \ell_2, \ell_5 \circ \ell_4, \ell_7 \circ \ell_6, \{\tilde{\ell}_i\}_{i=8}^{11}$, and ℓ_{12} are differentiable, and the rows of Jacobian are linearly independent, then UNet structure is invex, which is proven in the following. In the following analytical result, we prove the invexity of the UNet structure.

Theorem 12. *Assume $\mathcal{N}_\theta$ is the canonical UNet structure as illustrated in Figure 3 of the manuscript, and formally modeled in Equation* (23)*. Then, $\mathcal{N}_\theta$ is invex.*

Proof. Concretely, we have to show that all functions $\ell_1 \circ \ell, \ell_3 \circ \ell_2, \ell_5 \circ \ell_4, \ell_7 \circ \ell_6, \{\tilde{\ell}_i\}_{i=8}^{11}$, and ℓ_{12} are differentiable and the rows of their Jacobian are linearly independent. To that end, we proceed on a case-by-case basis.

1. The first aspect to mention is that UNet belongs to under-complete convolutional architectures [37] i.e. the number of filters do not exceed the dimension of the input. This implies that linearly independence within the convolutional layers is ensured.

2. The function $\ell_1 \circ \ell : \mathbb{R}^n \to \mathbb{R}^r$ with $r < n/2$ is defined as $(\ell_1 \circ \ell)(\boldsymbol{x}) = (g \circ f)(\boldsymbol{x})$. Here $f(\boldsymbol{x}) = \text{ReLU}(h(\boldsymbol{x}))$, where ReLU is applied element-wise, represents a transformation followed by a ReLU activation, where $h : \mathbb{R}^m \to \mathbb{R}^n$ models the consecutive composition of convolution layer, ReLU, and a convolution layer. The operator $g(\boldsymbol{z}) = \mathcal{M}(\boldsymbol{z})$ implements max pooling, with $g : \mathbb{R}^m \to \mathbb{R}^r$ reducing spatial dimensions.

   Following the methodology of Theorem 9, the Jacobian of f contains a block-diagonal structure contingent on the activation pattern of ReLU (i.e., inducing a binary diagonal matrix). Under batch normalization, half the entries of $h(\boldsymbol{x})$ are negative in expectation. Thus, row-rank of Jacobian of f, $\mathcal{J}_f(\boldsymbol{x})$, is $m/2$ in expectation for all $\boldsymbol{x}$.

   The max pooling operator g is a subsampling selector. Therefore, the Jacobian of g, $\mathcal{J}_g(\boldsymbol{z})$ is a binary matrix whose rows are linearly independent by construction.

   From the above analysis, we have that f and g satisfy the assumptions of Lemma 2, which leads to the conclusion that the Jacobian of $(\ell_1 \circ \ell)(\boldsymbol{x})$ is full-row-rank (i.e., all rows are linearly independent).

3. Notice that functions $\ell_3 \circ \ell_2, \ell_5 \circ \ell_4$, and $\ell_7 \circ \ell_6$ follows the same structure of the analyzed function $\ell_1 \circ \ell$, therefore, the Jacobian of all functions $\ell_3 \circ \ell_2, \ell_5 \circ \ell_4$, and $\ell_7 \circ \ell_6$ are full-row-rank (i.e., all rows are linearly independent).

4. Now we examine the mappings $\{\tilde{\ell}_i\}_{i=8}^{11}$, as defined in (24). The Jacobian matrix for each function in this set exhibits the structured form Concat $[\boldsymbol{I}, \mathcal{D}f(\boldsymbol{z})]$ where $\boldsymbol{I}$ denotes the identity matrix and $\mathcal{D}f(\boldsymbol{z})$ is the Jacobian of the second concatenated function specified in (24). As each $\tilde{\ell}_i$ constitutes a dimensionality-preserving mappings implying equivalence between input and output sizes –the inclusion of the identity $\boldsymbol{I}$ within the Jacobian maintains the number of linearly independent rows equal to the input dimension. And, this is a sufficient condition to establish the invexity of $\{\tilde{\ell}_i\}_{i=8}^{11}$ as shown in Corollary 1.

5. Lastly, the function ℓ_{12} is given by the consecutive composition of two Convolutional Layer with 64 filters with ReLU, and a Convolutional layer with 1 filter. As analyzed in second item when ReLU is used the Jacobian of ℓ_{12} will contain a block-diagonal structure contingent on the activation pattern of ReLU (i.e., inducing a binary diagonal matrix). Thus, row-rank of Jacobian of ℓ_{12} will be reduced by half of the input dimension in expectation. And, since just one filter is employed in last convolutional layer (output dimension is smaller than half of input) then by Lemma 2, the Jacobian of ℓ_{12} is full-row-rank.

Considering the above analysis, we can invoke Theorem 8 which ensures that all component functions of UNet structure $\mathcal{N}_\theta$ are invex with respect to the same η. Thus, the result holds. □

E Invexity for Generative Adversarial Network

In this section we study the invexity of the GAN structure illustrated in Figure 2. The importance of this analysis is that the size of the input is smaller than the output, but invexity remains attainable by using an embedding variable. Then, formally, the GAN $\mathcal{N}_\theta : \mathbb{R}^n \to \mathbb{R}^m$ in Figure 2 is mathematically given by

$$\mathcal{N}_\theta(\boldsymbol{x}) = \left(\ell_3 \circ \ell_2 \circ \cdots \circ \ell_1 \circ \hat{\ell}_{\boldsymbol{p}}\right)(\boldsymbol{x}), \tag{25}$$

where $\hat{\ell}_{\boldsymbol{p}}(\boldsymbol{x}) = \ell(\boldsymbol{x}) + \boldsymbol{p}$. In practical terms to implement the parameter $\boldsymbol{p}$ can be done as using the Pytorch

```
    self.p = nn.Parameter(torch.zeros(1, 784))

def forward(self, z):
    img = self.initlayer(z) + self.p
```

library.

In the following analytical result, we prove the invexity of the GAN structure.

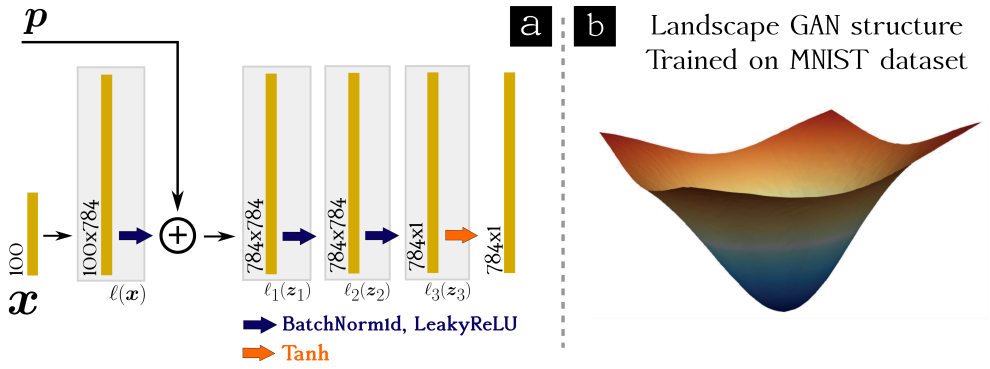


Figure 2: (a) We illustrate a structure of Generative Adversarial Network. (b) The loss landscape of GAN architecture. This landscape offers a visual validation of the invexity of the GAN i.e. exhibiting a connected global minimum.

Theorem 13. *Assume $\mathcal{N}_\theta$ is the GAN structure as illustrated in Figure 2, and formally modeled in Equation* (25)*. Then, $\mathcal{N}_\theta$ is invex.*

Proof. Concretely, we have to show that all functions $\hat{\ell}_{\boldsymbol{p}}, \ell_1, \ell_2$, and ℓ_3 are differentiable and the rows of their Jacobian are linearly independent. To that end, we proceed on a case-by-case basis.

1. Observe that the function $\hat{\ell}_{\boldsymbol{p}}$ is defined as $\hat{\ell}_{\boldsymbol{p}}(\boldsymbol{x}) = \ell(\boldsymbol{x}) + \boldsymbol{p}$. This means that the Jacobian of $\hat{\ell}_{\boldsymbol{p}}$ is given as $[\mathcal{D}\ell(\boldsymbol{x}) \;\; \boldsymbol{I}]$ where $\mathcal{D}\ell(\boldsymbol{x})$ is the Jacobian of $\ell(\boldsymbol{x})$, and $\boldsymbol{I}$ is the identity matrix. Then, it is clear that the Jacobian matrix $[\mathcal{D}\ell(\boldsymbol{x}) \;\; \boldsymbol{I}]$ is full-row-rank for any $\boldsymbol{x}$ (since identity matrix is horizontally concatenated).

2. The functions $\{\ell_i\}_{i=1}^3$ follows the form $\ell_i = \sigma(\boldsymbol{W}_i \boldsymbol{z}_i + \boldsymbol{b}_i)$ where the activation function σ (either LeakyReLU or Tanh) is applied element-wise, and $\boldsymbol{W}_i \in \mathbb{R}^{m_i \times n_i}$, where $m_i \leq n_i$. Since the number of rows m_i learnt weight matrices is smaller than n_i, and functions LeakyReLU/Tanh do not have stationary points, then the assumptions of Theorem 9, which leads to the conclusion that the Jacobians of $\{\ell_i\}_{i=1}^3$ is full-row-rank (i.e., all rows are linearly independent).

Considering the above analysis, we can invoke Theorem 8 which ensures that all component functions of the GAN structure $\mathcal{N}_\theta$ are invex with respect to the same η. Thus, the result holds. □

F Invexity for Vision Transformer Network

In this section we study the invexity of the Vision Transformer structure introduced in [38], and illustrated in Figure 3. Then, formally, the Vision Transformer $\mathcal{N}_\theta : \mathbb{R}^n \to \mathbb{R}^m$ is mathematically given by

$$\mathcal{N}_\theta(\boldsymbol{x}) = \left(\ell_r \circ \ell_{r-1} \circ \cdots \circ \ell_1 \circ \ell\right)(\boldsymbol{x}), \tag{26}$$

where $r+1$ is the number of consecutive stacked blocks in $\mathcal{N}_\theta$.

The importance of [38], is that it fundamentally reshaped computer vision paradigms by demonstrating that pure transformer architectures-previously dominant in natural language processing-could surpass state-of-the-art convolutional neural networks (CNNs) when applied directly to image classification. Its key innovation lay in tokenizing images into sequences of linearly embedded patches (treating image patches as "visual words"), enabling the application of standard transformer encoders without spatial inductive biases. Crucially, the work revealed that vision transformers achieve competitive performance only when pre-trained on sufficiently large datasets, after which they transfer exceptionally well to downstream tasks while exhibiting superior computational efficiency during training. By establishing that global self-attention mechanisms could outperform decades of CNN architectural refinements at scale, [38] catalyzed a paradigm shift toward attention-based vision models and laid the foundation for unified architectures across vision, language, and multimodal domains.

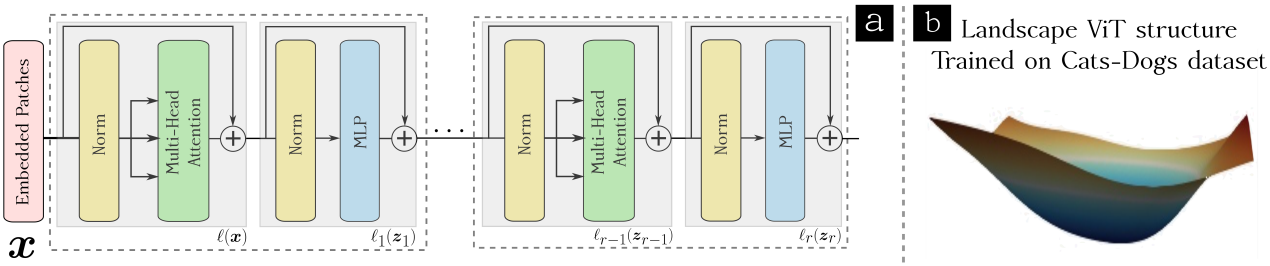


Figure 3: (a) We illustrate the canonical Vision Transformer architecture reported in [38]. (b) The loss landscape of Vision Transformer architecture, where $r = 6$, number of *heads* $= 16$, and dimension for MLP $= 256$ (following implementation in `https://github.com/lucidrains/vit-pytorch`). This landscape offers a visual validation of the invexity of the Vision Transformer i.e. exhibiting a connected global minimum.

In the following analytical result, we prove the invexity of the Vision Transformer structure.

Theorem 14. *Assume $\mathcal{N}_\theta$ is any Vision Transformer structure as illustrated in Figure 3, and formally modeled in Equation* (26)*. Then, $\mathcal{N}_\theta$ is invex.*

Proof. Concretely, we have to show that all functions $\ell, \ell_1, \ldots, \ell_{r-1}$, and ℓ_r are differentiable and the rows of their Jacobian are linearly independent. To that end, we proceed on a case-by-case basis.

1. Notice that the function $\ell(\boldsymbol{x})$ follows the form $\ell(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{x}$, where $f(\boldsymbol{x})$ contains the consecutive Norm, and Multi-head Attention transformations. This means that the Jacobian of $\ell(\boldsymbol{x})$ is given as $\boldsymbol{I} + \mathcal{D}f(\boldsymbol{x})$ where $\mathcal{D}f(\boldsymbol{x})$ is the Jacobian of $f(\boldsymbol{x})$, and $\boldsymbol{I}$ is the identity matrix. Since $f(\boldsymbol{x})$ entirely depends on the learned weights having complete freedom to choose its values, it is clear that $\boldsymbol{I} + \mathcal{D}f(\boldsymbol{x})$ is invertible, leading to a full-row-rank Jacobian matrix for $\ell(\boldsymbol{x})$ for any $\boldsymbol{x}$.

2. Observe that the remaining functions $\{\ell_i(\boldsymbol{z_i})\}_{i=1}^r$ follows the form $\ell_i(\boldsymbol{z}_i) = f_i(\boldsymbol{z}_i) + \boldsymbol{z}_i$, where $f_i(\boldsymbol{z}_i)$ contains the consecutive Norm, and Multi-head Attention or MLP transformations. Following an analogous process as in the previous item we conclude that the Jacobian matrix for all $\{\ell_i(\boldsymbol{z}_i)\}_{i=1}^r$ is full-row-rank for any $\boldsymbol{z}_i$.

Considering the above analysis, we can invoke Theorem 8 which ensures that all component functions of the Vision Transformer structure $\mathcal{N}_\theta$ are invex with respect to the same η. Thus, the result holds. □

G Experiments Details

We provide details of our experimental setup, including datasets, model architectures, training hyperparameters, and hardware/software environment.

G.1 Datasets

We conducted experiments on five publicly available benchmark datasets:

- **MNIST:** A widely used dataset of handwritten digits, comprising 70,000 grayscale images (28×28 pixels) of digits 0–9. The training set includes 60,000 images and the test set contains 10,000 images. This dataset was used to train MLP and GAN models.
- **CIFAR-10:** This dataset consists of 60,000 color images (32×32 pixels, RGB channels) equally distributed across 10 object classes, with 50,000 images for training and 10,000 for testing. CIFAR-10 was used to train canonical ResNet architectures.
- **ECG5000:** Sourced from the UCR Time Series Archive, ECG5000 contains 5,000 single-channel electrocardiogram (ECG) signals categorized into five classes. The dataset was used to evaluate ResNet variants on time-series data.
- **PASCAL VOC 2012:** A benchmark for semantic image segmentation, comprising 11,530 color (RGB) images with diverse resolutions (typically around 500×375 pixels) and pixel-level annotations for 21 object categories. The dataset was used to train UNet models.
- **Dogs vs. Cats:** This dataset includes 25,000 labeled color images (JPEG, RGB) of dogs and cats for training, and 12,500 unlabeled images for testing. Image sizes vary, typically ranging from 200×200 to 1050×771 pixels. The dataset was used to benchmark Vision Transformer models for image classification.

G.2 Computation of the Landscape

To compute the landscape of all neural networks studied in this paper we use the method reported in [39]. This method produces a landscape by analyzing the loss function used while training the neural network. The code for this method can be found in `https://github.com/tomgoldstein/loss-landscape`.

G.3 Model Architectures

We benchmarked four neural network families, described succinctly below:

MLP. A classic multilayer perceptron consisting of 3–5 fully connected hidden layers, each with 128–512 units, ReLU activations, and batch normalization.

ResNet101 [40].and SNDCDD [41] We employ the canonical 101-layer residual network as our deep convolutional backbone. Its bottleneck design (1×1, 3×3, 1×1 convolutions) and identity shortcuts facilitate gradient flow in very deep models. We use both the standard version (with skip connections, BatchNorm, and ReLU) and variants SNDCNN where it removes skip connections BatchNorm and replace activations with SELU to study robustness to architectural modifications.

UNet_base [42]. Our segmentation-oriented encoder–decoder comprises four downsampling stages and four upsampling stages. Each stage uses two 3×3 convolutions (with BatchNorm and ReLU) before pooling or transpose convolution, yielding feature maps that halve spatial resolution in the encoder and restore it in the decoder. The network has roughly 30 convolutional layers and peaks at 512 channels before the final 1×1 convolution to produce per-pixel class scores.

Vision Transformer [38]: The Vision Transformer structure analyzed here has in total $r = 6$ number of repeated blocks, number of *heads* $= 16$, and dimension for MLP $= 256$. We follow the following implementation in `https://github.com/lucidrains/vit-pytorch`.

Generative Adversarial Network: We train the structure that is reported in Figure 2.

G.4 Training Hyperparameters

All MLP, ResNet101 and UNet models (with or without skip connections) were trained with the Adam optimizer, a batch size of 8, and for 50 epochs. The rest of the models were trained using AdamW, a batch size of 64, and for 100 epochs, with a cosine annealing learning rate schedule and linear warmup. All other hyperparameters were set according to standard practice unless otherwise noted.

G.5 Hardware

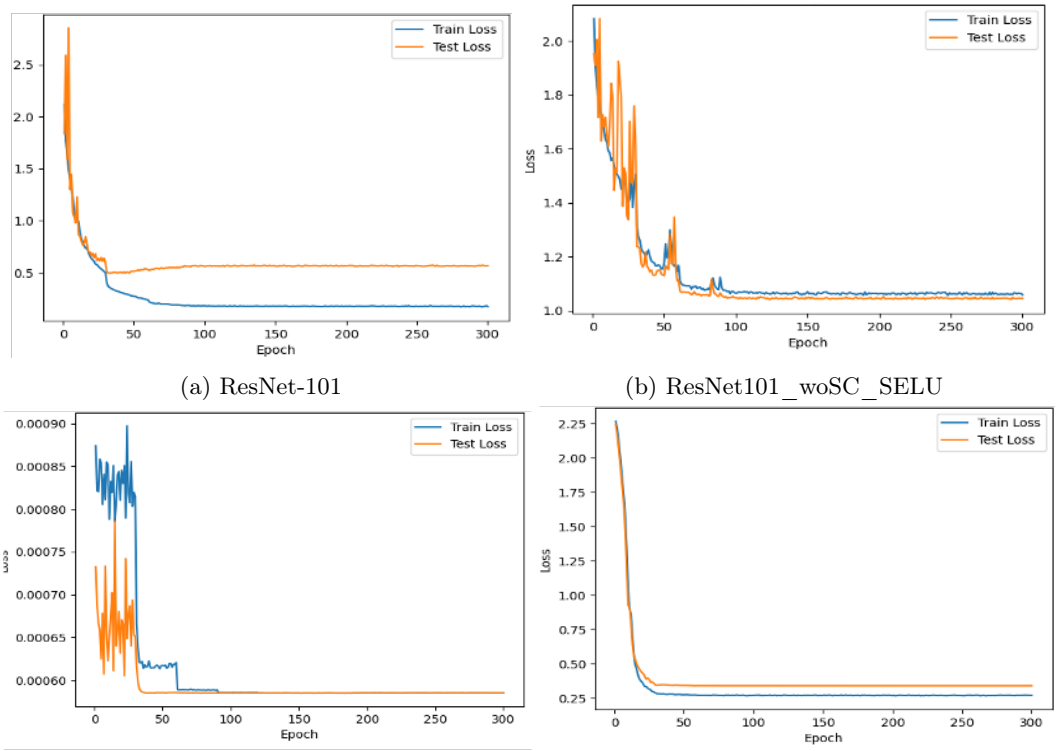
All experiments were conducted on a single NVIDIA A100 GPU using PyTorch. No distributed or multi-GPU training was employed.

G.6 Additional experiments on Deep Architectures

To systematically examine the impact of architectural components on the trainability and stability of deep convolutional networks, we conducted ablation experiments on both image (CIFAR-10) and sequential (ECG5000) datasets. The experiments compared: (i) a baseline ResNet-101 with skip connections, BatchNorm, and ReLU activations; (ii) a variant with skip connections removed and SELU activations (BatchNorm retained); (iii) a further ablated model with both skip connections and BatchNorm eliminated, and all activations set to SELU, closely matching the SNDCNN architecture [41]; and (iv) SNDCNN itself evaluated on 1D ECG5000 data.

Figure 4 summarizes the training and test loss curves for all variants. The baseline ResNet-101 on CIFAR-10 [Fig.4(a)] achieves stable convergence with a moderate gap between training and test loss, and reaches a final test accuracy of 82.7% (Table4). Removing skip connections and switching to SELU activation [Fig.4(b)] results in less stable training and a substantial drop in accuracy (62.7%). Further eliminating BatchNorm [Fig.4(c)] leads to a complete collapse: while the loss decreases, the network fails to generalize, producing only chance-level accuracy (10%), consistent with prior observations that self-normalizing networks alone are inadequate for high-dimensional image tasks.

In contrast, applying the same SNDCNN-style architecture to the 1D ECG5000 dataset [Fig. 4(d)] yields smooth convergence of both training and test loss, and a high test accuracy of 91.7%. These results underscore the critical role of skip connections and BatchNorm in training deep CNNs on complex image data, and highlight that self-normalizing networks (with SELU) can remain highly effective for certain 1D sequential domains.



(a) ResNet-101 (b) ResNet101_woSC_SELU

(c) ResNet101_woSC&BN_SELU (d) SNDCNN

Figure 4: Training and test loss curves for various deep convolutional architectures. (a) Baseline ResNet-101 (with skip connections, BatchNorm, and ReLU) on CIFAR-10 exhibits stable convergence and a clear train-test loss gap. (b) ResNet101_woSC_SELU (skip connections removed, BatchNorm retained, SELU activation) maintains trainability on CIFAR-10, though with reduced stability. (c) ResNet101_woSC&BN_SELU (both skip connections and BatchNorm removed, SELU activation; SNDCNN-like) on CIFAR-10 achieves low loss but fails to generalize beyond chance level. (d) SNDCNN on ECG5000 (1D data) demonstrates smooth and consistent convergence for both train and test loss, confirming trainability in sequential signal domains.

In summary, these ablation experiments demonstrate that, while self-normalizing architectures can be trained and perform well for low-dimensional sequential signals, both skip connections and BatchNorm are indispensable

Table 4: Ablation study: final test accuracy (%) of each model on CIFAR-10 and ECG5000, with skip connection, BatchNorm, and activation settings explicitly indicated.

Model	Dataset	Skip Conn.	BatchNorm	Activation	Final Test Accuracy (%)
ResNet101	CIFAR-10	✓	✓	ReLU	82.7
ResNet101_woSC_SELU	CIFAR-10	×	✓	SELU	62.7
ResNet101_woSC&BN_SELU	CIFAR-10	×	×	SELU	10.0
SNDCNN	ECG5000	×	×	SELU	91.7

for enabling deep convolutional networks to generalize on high-dimensional image data.

Analysis: It is essential to note that trainability, as studied in this work, addresses a specific yet fundamental aspect of deep learning: why optimization succeeds, rather than whether the model itself is adequate. An example of this is ResNet structure without skip connections and replacing ReLU and SELU (as analyzed above). It is clear that this variant of ResNet is trainable (it is invex), but it is not enough to enabling deep convolutional networks to generalize on high-dimensional image data.

This distinction highlights the role of invexity as a structural condition for optimization, independent of dataset quality, task complexity, or model expressiveness. Specifically, our results, as presented in Propositions 11 and 12, refer to the ability of ResNet and UNet structures to reach a global optimum of their loss function —not necessarily to the network's success in solving a given task. Invexity in the neural network structure ensures that, when training succeeds, it does so in the optimal way according to the model's objective. However, this does not guarantee that the trained neural network is appropriate for the intended application.

References

[1] V. Kunc and J. Kléma, "Three decades of activations: A comprehensive survey of 400 activation functions for neural networks," *arXiv preprint arXiv:2402.09092*, 2024.

[2] J. Sharma, "Evaluating CNN with Oscillatory Activation Function," *arXiv preprint arXiv:2211.06878*, 2022.

[3] A. Bagirov, N. Karmitsa, and M. M. Mäkelä, *Introduction to Nonsmooth Optimization: theory, practice and software*. Springer, 2014.

[4] T. W. Reiland, "Nonsmooth invexity," *Bulletin of the Australian Mathematical Society*, vol. 42, no. 3, pp. 437–446, 1990.

[5] S. K. Mishra and G. Giorgi, *Invexity and optimization*, vol. 88. Springer Science & Business Media, 2008.

[6] S. Pinilla and J. Thiyagalingam, "Global optimality for non-linear constrained restoration problems via invexity," in *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[7] G. S. da S. Gomes, T. B. Ludermir, and L. M. Lima, "Comparison of new activation functions in neural network for forecasting financial time series," *Neural Computing and Applications*, vol. 20, pp. 417–439, 2011.

[8] M. Kaytan, İ. B. Aydilek, C. Yeroğlu, and A. Karci, "Sigmoid-Gumbel: Yeni Bir Hibrit Aktivasyon Fonksiyonu," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 11, no. 1, pp. 29–45, 2022.

[9] M. Klimek and M. Perelstein, "Neural network-based approach to phase space integration," *SciPost Physics*, vol. 9, no. 4, p. 053, 2020.

[10] W. Duch and N. Jankowski, "Survey of neural transfer functions," *Neural computing surveys*, vol. 2, no. 1, pp. 163–212, 1999.

[11] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[12] K. Wong, R. Dornberger, and T. Hanne, "An analysis of weight initialization methods in connection with different activation functions for feedforward neural networks," *Evolutionary Intelligence*, vol. 17, no. 3, pp. 2081–2089, 2024.

[13] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021.

[14] H. H. Chieng, N. Wahid, P. Ong, and S. R. K. Perla, "Flatten-T swish: a thresholded ReLU-Swish-like activation function for deep learning," *arXiv preprint arXiv:1812.06247*, 2018.

[15] A. De Brebisson and P. Vincent, "An exploration of softmax alternatives belonging to the spherical loss family," *arXiv preprint arXiv:1511.05042*, 2015.

[16] Y. Koçak and G. Ü. Şiray, "New activation functions for single layer feedforward neural network," *Expert Systems with Applications*, vol. 164, p. 113977, 2021.

[17] T. T. Sivri, N. P. Akman, and A. Berkol, "Multiclass classification using arctangent activation function and its variations," in *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–6, IEEE, 2022.

[18] L. B. Godfrey and M. S. Gashler, "A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks," in *2015 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K)*, vol. 1, pp. 481–486, IEEE, 2015.

[19] Q. Fan, F. H. Hou, F. Shi, *et al.*, "Bent identity-based CNN for image Denoising," *Journal of Applied Science and Engineering*, vol. 23, no. 3, pp. 547–554, 2020.

[20] B. Carlile, G. Delamarter, P. Kinney, A. Marti, and B. Whitney, "Improving deep learning by inverse square root linear units (ISRLUs)," *arXiv preprint arXiv:1710.09967*, 2017.

[21] Y. Liu, J. Zhang, C. Gao, J. Qu, and L. Ji, "Natural-logarithm-rectified activation function in convolutional neural networks," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, pp. 2000–2008, IEEE, 2019.

[22] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International conference on machine learning*, pp. 1319–1327, PMLR, 2013.

[23] S. K. Roy, S. Manna, S. R. Dubey, and B. B. Chaudhuri, "LiSHT: Non-parametric linearly scaled hyperbolic tangent activation function for neural networks," in *International Conference on Computer Vision and Image Processing*, pp. 462–476, Springer, 2022.

[24] H. Zhu, H. Zeng, J. Liu, and X. Zhang, "Logish: A new nonlinear nonmonotonic activation function for convolutional neural network," *Neurocomputing*, vol. 458, pp. 490–499, 2021.

[25] M. Kaytan, I. B. Aydilek, and C. Yeroğlu, "Gish: a novel activation function for image classification," *Neural Computing and Applications*, vol. 35, no. 34, pp. 24259–24281, 2023.

[26] Z. Wu, H. Yu, L. Zhang, and Y. Sui, "The Adaptive Quadratic Linear Unit (aqulu): Adaptive Non-Monotonic Piecewise Activation Function," *Tehnički vjesnik*, vol. 30, no. 5, pp. 1469–1485, 2023.

[27] T. T. Sivri, N. P. Akman, and A. Berkol, "The Impact of Irrationals on the range of Arctan Activation Function for Deep Learning Models," *International Journal of Engineering Technologies IJET*, vol. 9, no. 3, pp. 89–101, 2023.

[28] J. T. Barron, "Squareplus: A softplus-like algebraic rectifier," *arXiv preprint arXiv:2112.11687*, 2021.

[29] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," *arXiv preprint arXiv:1511.07289*, 2015.

[30] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in neural information processing systems*, vol. 31, 2018.

[31] L. Eidnes and A. Nøkland, "Shifting mean activation towards zero with bipolar activation functions," *arXiv preprint arXiv:1709.04054*, 2017.

[32] O. L. Mangasarian, *Nonlinear Programming*. Society for Industrial and Applied Mathematics, 1994.

[33] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *International conference on machine learning*, pp. 146–155, PMLR, 2017.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Conference on learning theory*, pp. 2306–2327, PMLR, 2020.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[37] J. M. J. Valanarasu, V. A. Sindagi, I. Hacihaliloglu, and V. M. Patel, "Kiu-net: Overcomplete convolutional architectures for biomedical image and volumetric segmentation," *IEEE Transactions on Medical Imaging*, vol. 41, no. 4, pp. 965–976, 2021.

[38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[39] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Advances in neural information processing systems*, vol. 31, 2018.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[41] Z. Huang, T. Ng, L. Liu, H. Mason, X. Zhuang, and D. Liu, "Sndcnn: Self-normalizing deep cnns with scaled exponential linear units for speech recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6854–6858, 2020.

[42] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.