EFFECTIVE RANK AND THE STAIRCASE PHENOMENON: NEW INSIGHTS INTO NEURAL NETWORK TRAINING DYNAMICS*

JIANG YANG†, YUXIANG ZHAO‡, AND QUANHUI ZHU‡

Abstract. In recent years, deep learning, powered by neural networks, has achieved widespread success in solving high-dimensional problems, particularly those with low-dimensional feature structures. This success stems from their ability to identify and learn low dimensional features tailored to the problems. Understanding how neural networks extract low-dimensional features from high-dimensional data remains a fundamental challenge in deep learning theory. In this work, we introduce the concept of 'effective rank' to measure the linear independence of the basis functions represented by the neurons in the final hidden layer. Through extensive numerical experiments, we have discovered a notable phenomenon: the effective rank increases progressively during training, exhibiting a staircase-like pattern, while the loss function concurrently decreases. We refer to this observation as the 'staircase phenomenon'. In addition, for deep neural networks, we rigorously provide a theoretical explanation for this phenomenon by establishing the negative correlation between the loss function and effective rank, demonstrating that the lower bound of the loss function decreases with increasing effective rank. Therefore, to achieve a rapid descent of the loss function, it is critical to promote the swift growth of effective rank. We also evaluate several advanced learning methodologies and find that they can indeed accelerate the training process by quickly increasing the effective rank.

Key words. deep neural network, training dynamics, staircase phenomenon, effective rank.

MSC codes. 68Q32, 68T07

1. Introduction.

Deep neural networks (DNNs) have exhibited remarkable performance across a wide range of fields, including computer vision, natural language processing, and computational physics, due to their extraordinary representation capabilities. However, the training process of DNNs remains highly complex and, in many respects, poorly understood. Developing a comprehensive understanding of the training mechanisms is critical for the design, optimization, and interpretability of these models.

There are various approaches attempting to explain the mechanism of training dynamics in neural networks. Visualization-based methods, such as those in [39, 21], analyze the hierarchical formation of feature maps in convolutional networks and investigate how network architecture influences the geometry of the loss landscape during training. The impact of flat and sharp minima on generalization performance has been extensively studied in [13, 5, 19], providing insights into the relationship between the optimization landscape and generalization behavior. The neural tangent kernel (NTK) theory provides a rigorous framework for analyzing the training dynamics of wide neural networks in the infinite-width limit. Jacot et al. [18] introduce the NTK to describe the training dynamics as a convergent kernel in function space, offering valuable insights into how networks evolve during training. Extensions of the NTK framework in [32, 1, 4] further examine the generalization behavior of neural networks across various contexts. Another important perspective comes from the frequency principle (or spectral bias), suggesting that neural networks tend to learn low-frequency patterns during the early stages of training [34, 27, 36, 35]. This

*Submitted to the editors DATE.

†Department of Mathematics, SUSTech International Center for Mathematics & National Center for Applied Mathematics Shenzhen (NCAMS), Southern University of Science and Technology, Shenzhen, China (yangj7@sustech.edu.cn).

‡Department of Mathematics, Southern University of Science and Technology, Shenzhen, China (12131241@mail.sustech.edu.cn, 12131244@mail.sustech.edu.cn).

observation demonstrates that, when the training process of the neural network is projected into the spectral domain, the number of effective frequencies exhibits an increasing trend over time.

A multilayer perceptron (MLP) neural network can be constructed as

$$\begin{cases} y_0 = x, \\ y_{k+1} = \sigma(W_k y_k + b_k), \quad k = 0, \cdots, L-1, \\ y = \beta \cdot y_L, \end{cases} \tag{1.1}$$

where $x \in \mathbb{R}^d$, $y_k \in \mathbb{R}^n$, and $W_0 \in \mathbb{R}^{n\times d}, W_k \in \mathbb{R}^{n\times n}, b_k, \beta \in \mathbb{R}^n$ are trainable parameters. Denoting $\theta = \{W_k, b_k\}_{k=0}^{L-1}$, the output of the neural network can alternatively be expressed in the form

$$y(x;\theta) = \sum_{j=1}^{n} \beta_j \phi_j(x;\theta).$$

We decompose the neural network into two parts: the **neuron functions** $\{\phi_j\}_{j=1}^n$ correspond to the neurons in the last hidden layer, and the coefficients $\{\beta_j\}_{j=1}^n$ are the weights in the output layer. This work focus on examining neural networks from the perspective of traditional computational mathematics. Specifically, when the output of a neural network is expressed as a linear combination of the neurons in the last hidden layer, these neurons can be interpreted as a set of basis functions. [11, 10] show that ReLU-activated deep neural networks can reproduce all linear finite element functions and [23, 24, 30] provide algorithms to combine classical finite element methods with neural networks. As discussed in [16, 26], randomly generated neuron functions in shallow neural networks also exhibit sufficient representation ability. A detailed analysis of the coefficient matrices associated with the random features, particularly in terms of the distribution of their singular values, has been provided in [3]. The decay rate of the eigenvalues of the Gram matrix of a two-layer neural network with ReLU activation under general initialization has been analyzed in [40]. Its further numerical study suggests that smoother activation functions lead to faster spectral decay of the Gram matrix. These findings underscore the connection between traditional computational methods and deep learning frameworks, motivating a deeper mathematical exploration of neural network representations.

In this work, we focus on understanding how neuron functions evolve during the training dynamics of neural networks. Motivated by prior researches, we investigate the behavior of the singular values of the mass matrix associated with the neuron functions during the training process. To formalize the observations, we introduce the concept of **ϵ-rank** (effective rank) for a set of functions (see as Definition 2.5), which quantitatively represents the number of effective features in the network.

Our key finding in this paper is the identification of a novel phenomenon concerning the ϵ-rank of neuron functions, stated as follows:

***Staircase phenomenon**: In training dynamics, the loss function often decreases rapidly along with a significant growth of ϵ-rank of neurons, and the evolution of the ϵ-rank over time resembles a staircase-like pattern.*

As shown in the Figure 1, this increase in ϵ-rank occurs in a stepwise fashion, closely resembling the structure of a staircase. Specifically, under standard parameter initialization, neuron functions initially exhibit low linear independence. Throughout the training process, variations in the loss function are closely linked to changes in the linear independence of these neuron functions. When the decline in the loss function

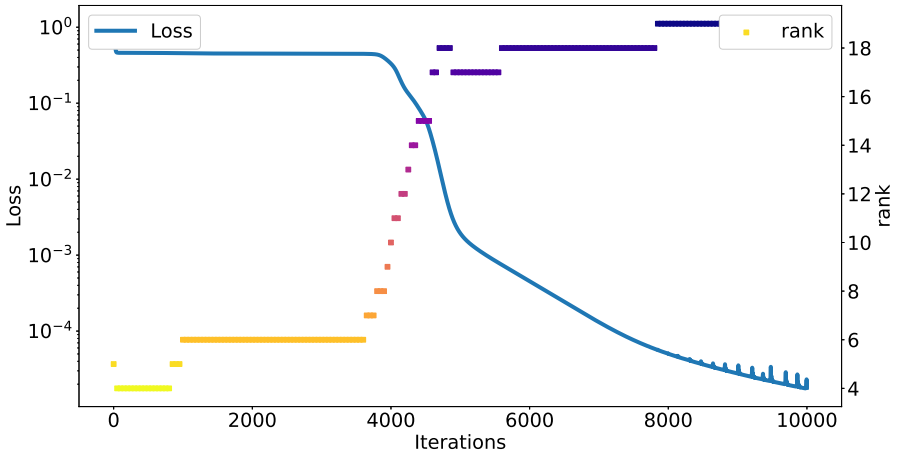


Fig. 1: Staircase phenomenon of neuron functions in training dynamics.

reaches a plateau, the growth in linear independence tends to stabilize. Conversely, when the linear independence increases significantly, the loss experiences a rapid descent. We theoretically prove that the lower bound of the loss function decreases as the ϵ-rank increases. This theoretical result holds for general deep neural networks. Consequently, achieving a sufficiently large ϵ-rank is essential for ensuring a significant reduction in the loss function.

This naturally raises the question: how can a set of highly linearly independent neuron functions be learned efficiently? To address this, we identify several strategies from existing techniques that enable the neuron functions with a high ϵ-rank at the early stage of training. Specifically, by designing appropriate initialization schemes and selecting well-suited network architectures, it is possible to ensure that the neuron functions exhibit high linear independence from the outset. Such approaches significantly accelerate the training process or improve the accuracy.

The main contributions of this work are summarized as follows:

1. The introduction of effective rank provides a novel perspective for understanding the training dynamics of neural networks, revealing a staircase phenomenon. This phenomenon is observed universally across various tasks, including function fitting, handwriting recognition, and solving partial differential equations.
2. We prove that the loss function of deep neural networks has a lower bound related to the ϵ-rank. This bound decreases as the ϵ-rank increases, thereby providing a theoretical explanation for the staircase phenomenon. Our analysis concludes that a sufficiently high ϵ-rank is a necessary condition for achieving a significant reduction in the loss function.
3. We investigate how ϵ-rank evolves in existing advanced methodologies. Numerical examples show that these methodologies can construct neuron functions with a high ϵ-rank rapidly, thereby significantly accelerating the training dynamics and improving model accuracy.

The rest of this paper is organized as follows. In section 2, the core concept of

ϵ-rank is defined through eigenvalues of the Gram matrix, and the staircase phenomenon is examined across various tasks and settings. A lower bound on the loss function with respect to the ϵ-rank is given in section 3, which provides the theoretical explanation of the staircase phenomenon. The evolution of ϵ-rank under existing methods is tested through numerical experiments in section 4, which include initialization methods, specific network structures and partial of unity technique in the random feature method. Concluding remarks are given in the last section.

2. Staircase Phenomenon.

2.1. Preliminary.

Before presenting the staircase phenomenon, we introduce several definitions that are extensions of linear algebra.

DEFINITION 2.1. *The n functions $f_1(x), \cdots, f_n(x)$ are **linearly dependent** in domain Ω if, there exists $c_1, c_2, \cdots, c_n \in \mathbb{R}$ not all zero s.t.*

$$c_1 f_1(x) + c_2 f_2(x) + \cdots + c_n f_n(x) = 0, \qquad \forall x \in \Omega. \tag{2.1}$$

If the functions are not linearly dependent, they are said to be **linearly independent**. For given n functions $f_1(x), \cdots, f_n(x)$ in $L^2(\Omega)$, the mass matrix M is defined as follows

$$(M)_{ij} := \int_\Omega f_i(x) f_j(x) \mathrm{d}x, \quad 1 \le i, j \le n. \tag{2.2}$$

By c'Mc =\sum g_ig_j

Obviously M is symmetric positive semi-definite.

LEMMA 2.2. *$f_1(x), \cdots, f_n(x)$ in are linearly independent if and only if the rank of the mass matrix $r(M) = n$.*

While the concept of linear dependence is well-defined in linear algebra, few functions in real-world applications strictly satisfy the condition specified in (2.1). Consequently, a more practical and applicable indicator, extending beyond the framework of (2.1), is required.

DEFINITION 2.3. *The n functions $f_1(x), \cdots, f_n(x)$ in are **ϵ-linearly dependent** in domain Ω for some $\epsilon \ge 0$ if, there exists $c_1, c_2, \cdots, c_n \in \mathbb{R}$ with $\|\mathbf{c}\|^2 = 1$ s.t.,*

$$\|c_1 f_1 + c_2 f_2 + \cdots + c_n f_n\|^2 \le \epsilon. \tag{2.3}$$

*Otherwise they are said to be **ϵ-linearly independent**.*

Here $\|\cdot\|$ is the short note of L^2 norm, $\|u\| = \sqrt{\langle u, u\rangle} = \left(\int_\Omega u^2 \mathrm{d}x\right)^{\frac{1}{2}}$. It is easy to check that $f_1(x), \cdots, f_n(x)$ are ϵ-linearly independent if and only if the minimum eigenvalue of M satisfies $\lambda_{\min}(M) > \epsilon$. In the following, we intend to extend these concepts of linear independence to the neuron functions of a neural network.

DEFINITION 2.4. *For a given neural network $u(x;\theta)$ defined on $\Omega \subset \mathbb{R}^d$ as*

$$u(x;\theta) = \sum_{j=1}^{n} \beta_j \phi_j(x;\theta),$$

the Gram matrix M_u is defined as below

$$(M_u)_{ij} = \int_\Omega \phi_i(x;\theta)\phi_j(x;\theta)\mathrm{d}x.$$

Straightforwardly, we have the following definition of ϵ-effective rank, ϵ-rank for short.

DEFINITION 2.5. *The* ϵ***-rank*** *of a neural network* $u(x,\theta)$*, associated with its Gram matrix* M_u*, is defined as*

$$r_\epsilon(M_u) := |\{\lambda(M_u) > \epsilon\}|, \tag{2.4}$$

where $|\cdot|$ *denotes the cardinal number of a set.*

The standard definitions of linear dependence and rank can be regarded as special cases corresponding to $\epsilon = 0$. For simplicity, the linear independence and rank mentioned in experiments refer to ϵ-linear independence and ϵ-rank, respectively.

2.2. Staircase Phenomenon. With the necessary groundwork established, we now focus on how the ϵ-rank of the neuron basis functions evolves throughout the training process of the neural network. Across various tasks, including function fitting, handwriting recognition, and solving partial differential equations, we consistently observe the following phenomenon.

Staircase phenomenon*: In training dynamics, the loss function often decreases rapidly along with a significant growth of* ϵ*-rank of neurons, and the evolution of the* ϵ*-rank over time resembles a staircase-like pattern.* The ϵ-rank, as defined in Definition 2.5, quantifies the linear independence or effective features of neuron functions in the last hidden layer. We experimentally demonstrate this phenomenon under different settings. To illustrate this phenomenon, we first consider a function fitting problem, which is one of the most fundamental tasks in neural networks.

Example 2.6 (Function fitting). Consider the target function composed of multiple frequency components, defined as

$$f(x) = \cos x + \cos 2x + \cos 30x. \tag{2.5}$$

The computational domain is $\Omega = [-1, 1]$, and the mean square error is used as the loss function.

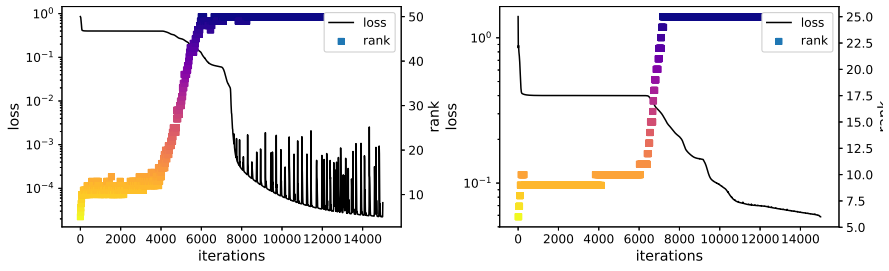
This example consists of three distinct experimental settings:

(i) (Figure 2) Investigate the performance of neural networks with varying width and depth. The network configurations are as follows: (a) $L = 2, n = 50$. (b) $L = 2, n = 25$. (c) $L = 4, n = 50$. (d) $L = 4, n = 25$.

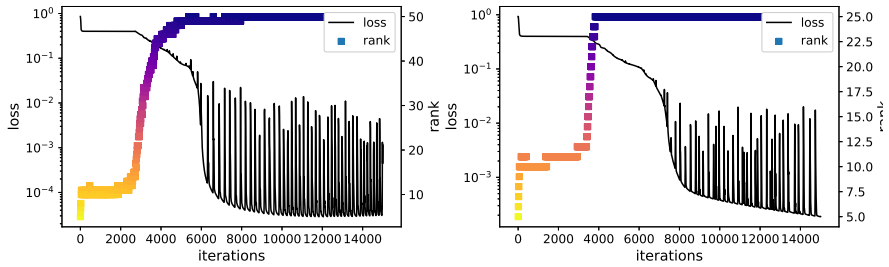
(ii) (Figure 3) Analyze the ϵ-rank of the neuron functions across different layers for a fixed network width of $n = 50$ and depth of $L = 4$.

(iii) (Figure 4) Test neural networks with the following different activation functions:
- ReLU: $\sigma(x) = \max(x, 0)$.
- ELU: $\sigma(x) = x$, if $x > 0$, and $\sigma(x) = \alpha(e^x - 1)$, if $x < 0$.
- Cosine: $\sigma(x) = \cos(x)$.
- Hyperbolic tangent: $\sigma(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

Under different width and depth, the training results are presented in Figure 2. The black line represents the loss function, while the scatters correspond to the ϵ-rank of the mass matrix. When the linear independence of the neuron functions does not increase, the loss function stagnates for an extended period. However, once the neuron functions become fully linearly independent, the loss function rapidly decreases to a lower level, as shown in Figure 2(a), (c) and (d). Subfigure (b) demonstrates that



(a) $L = 2, n = 50$ (b) $L = 2, n = 25$



(c) $L = 4, n = 50$ (d) $L = 4, n = 25$

Fig. 2: (Example 2.6) Staircase phenomenon under different width and depth. The black line is the mean square error and the gradient ladder is the ϵ-rank.

shallow and narrow networks fail to approximate the target function efficiently under insufficient training. Another notable observation from Figure 2(d) is that, after attaining full rank, the loss function experiences a further sharp decline. This phase highlights a further unknown step in the learning process.

Additionally, we plot the ϵ-rank of neurons in each hidden layer for the case of $L = 4, n = 50$ in Figure 3. The staircase phenomenon is observed in most hidden layers. The results also indicate that shallow neurons exhibit fewer features compared to deeper neurons. Within the same network, the ϵ-rank of deeper neurons is consistently larger than that of shallow neurons. This observation suggests that deeper layers not only retain the linear independence of the preceding shallow layers but also further increase the complexity of the neuron functions, enabling the network to represent more intricate features. Notably, neurons in the first hidden layer fundamentally differ from that of deep layers. In the first layer, the neuron functions act as a set of basis functions that are directly formed by the activation function, without the composite transformation process that characterizes deeper networks.

The appearance of staircase phenomenon is independent of the choice of activation function. Figure 4 illustrates the evolution of the ϵ-rank during the training processes under several commonly used activation functions. The results show that while its behavior differs depending on the specific activation function used, staircase phenomenon is present across all activation functions. It can be observed that the upward trend and the peak value of the ϵ-rank vary across different activation functions. This is because neural network function classes constructed with different

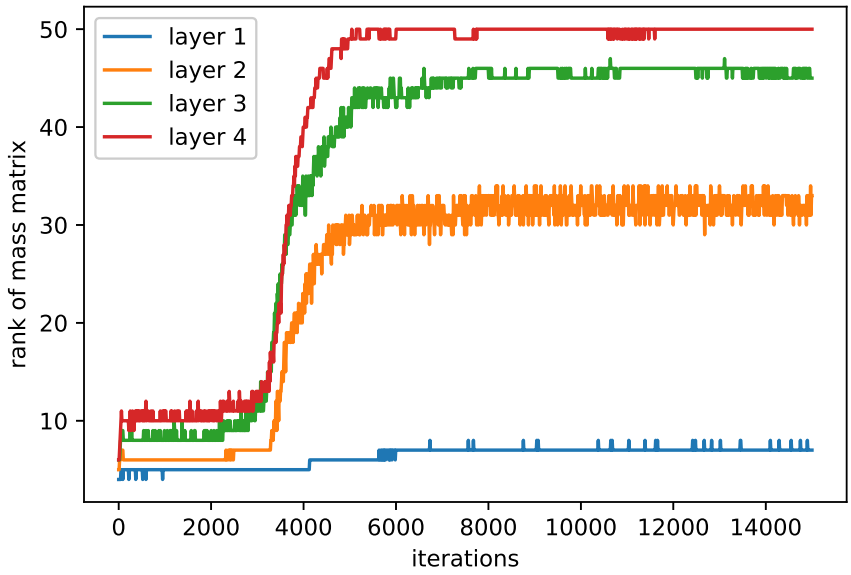


Fig. 3: (Example 2.6) The staircase phenomenon of different hidden layers in the same network

activation functions exhibit distinct approximation capabilities with respect to the target function.

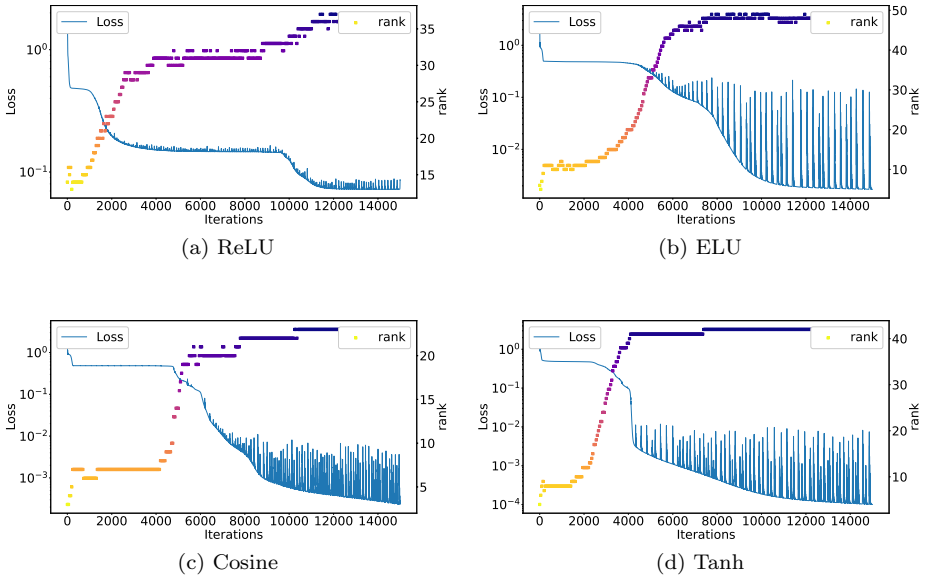
Moreover, this phenomenon not only occurs in function fitting, but also in handwriting recognition and solving partial differential equations (subsection 4.1).

Example 2.7 (Handwriting recognition, Figure 5). MNIST is a widely used database of handwritten digits for training various image processing systems. The dataset contains 70,000 grayscale images of handwritten digits (0 through 9), each with a resolution of 28x28 pixels. Thus, the dimension of the input is $d = 784$. In this example, the cross entropy loss function is employed for training.

To address errors caused by high-dimensional numerical integration, we introduce an intermediate hidden layer with 10 neurons, using the neurons in this layer as sampling points for integration. As illustrated in Figure 5, the staircase phenomenon is evident in the handwriting recognition task, demonstrating that this phenomenon can also be observed in high-dimensional problems. This observation underscores the representational power of neural networks, which can effectively construct high-dimensional feature spaces, that is notably challenging in traditional scientific computing.

To gain a clearer understanding of the training mechanism, it is necessary to investigate why the staircase phenomenon occurs. Many widely used initialization methods do not ensure that the neuron functions are fully linearly independent after initialization. Therefore, the first stage of training involves separating the neuron functions post-initialization. This is when the staircase phenomenon typically occurs.

Existing widely used initialization methods, like Xavier or Glorot initialization



(a) ReLU (b) ELU

(c) Cosine (d) Tanh

Fig. 4: (Example 2.6) Staircase phenomenon under different activation functions

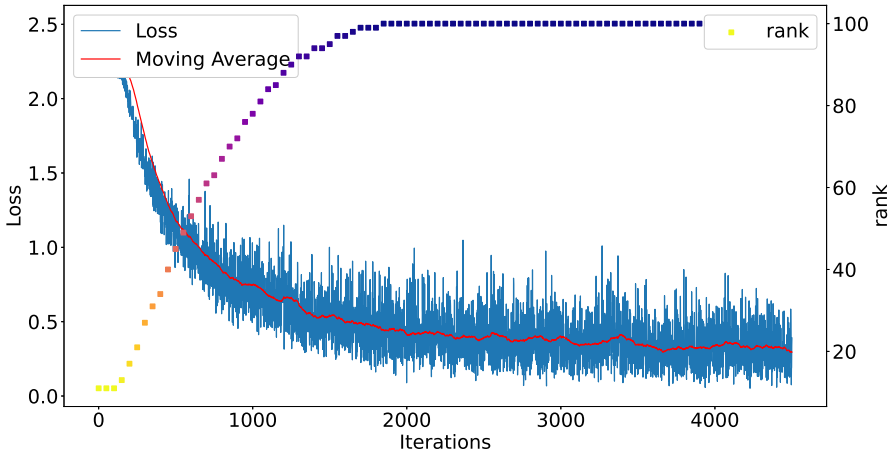


Fig. 5: (Example 2.7) Ladders phenomenon in handwriting recognition.

method [9], initialize the weights and biases as $w_j, b_j \sim U(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$. Consider a simple neuron function $\phi(x) = \sigma(wx + b)$, $x \in [-1, 1]$, with the hyperbolic tangent activation function $\sigma(x) = \tanh(x)$. Since $\sigma'(0) = 1$, the activation function behaves

approximately as a linear function near zero:

$$\phi(x) = wx + b - \frac{(wx+b)^3}{3} + o((w+b)^3) \sim wx + b.$$

As the linear combination of linear functions is also linear, the ϵ-rank of $\{\phi_i\}_{i=1}^n$ after initialization is approximately two.

This explains why, during the training process of function fitting, neural networks often experience a plateau in the loss function that takes time to overcome. Initially, this plateau arises because the initialized basis functions are not fully linearly independent. This phase can be interpreted as feature extraction. From our perspective, at specific stages during training, this process requires the separation of neuron functions and an increase in their ϵ-rank.

3. Theoretical Explanation of the Staircase Phenomenon. In this section, we demonstrate the theoretical significance of ϵ-rank and provide an explanation of the staircase phenomenon. For well-posed problems, when the solution lies outside the class of neural network functions, we demonstrate that an increase in ϵ-rank during the training process is a necessary condition for a reduction in the loss function.

It is trivial that if f is a linear combination of $f_1, f_2, \cdots, f_n$ with $r(M) = p$, then f can be rewritten as a linear combination of p of them. Similarly, if $r_\epsilon(M) = p$, a comparable result holds. We begins with the following useful lemma constructed in [14].

LEMMA 3.1. *[14] Let $Q \in \mathcal{O}_{n,p}$ with $p \le n$, and $\{Q_1, Q_2, \cdots, Q_k\}$ is the set of all p-by-p sub-matrices of Q where $k = \binom{n}{p}$. Define a vector in $\mathbb{R}^k$ by $\sigma(Q) = (\sigma_{\min}(Q_1), \sigma_{\min}(Q_2), \cdots, \sigma_{\min}(Q_k))^T$. Then*

$$\inf_{Q \in \mathcal{O}_{n,p}} (\|\sigma(Q)\|_\infty) \ge \frac{1}{\sqrt{p(n-p) + \min(p, n-p)}}.$$

THEOREM 3.2. *If $f = \sum_{j=1}^n \beta_j f_j$ with $\|\beta\|^2 \le C$, $r_\epsilon(M_f) = p$ for some $\epsilon \ge 0$ and positive integer $p \le n$, then after a reorder of set $\{f_1, \cdots, f_n\} = \{\tilde{f}_1, \cdots, \tilde{f}_n\}$ if necessary, f can be approximated by $\tilde{f} = \sum_{j=1}^p \tilde{\beta}_j \tilde{f}_j$ with $\|f - \tilde{f}\|^2 \le C(p+1)(n-p)^2\epsilon$.*

Proof. Consider the spectral decomposition $M = Q\Lambda Q^T$, Q is orthogonal, $\Lambda = \mathrm{diag}([\lambda_1, \lambda_2, \cdots, \lambda_n])$ is diagonal with eigenvalues $\lambda_1 \ge \cdots \ge \lambda_p > \epsilon \ge \lambda_{p+1} \ge \cdots \ge \lambda_n \ge 0$. For simplicity, we denote $\beta = \begin{bmatrix}\beta_1 & \cdots & \beta_n\end{bmatrix}^T$ and $F = \begin{bmatrix}f_1 & \cdots & f_n\end{bmatrix}^T$, thus $f = \beta^T F$. Let P be a permutation matrix, and denote

$$PQ = \begin{array}{c} \\ p \\ n-p \end{array} \begin{array}{c} \begin{array}{cc} p & n-p \end{array} \\ \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \end{array}$$

as a partitioning of the matrix PQ. Now we construct an approximation of f by $\tilde{f} = \tilde{\beta}^T \tilde{F}$, where

$$\begin{aligned} \tilde{\beta} &= \begin{bmatrix} I_p & 0 \end{bmatrix} P\beta - V_{12}V_{22}^{-1} \begin{bmatrix} 0 & I_{n-p} \end{bmatrix} P\beta, \\ \tilde{F} &= \begin{bmatrix} I_p & 0 \end{bmatrix} PF. \end{aligned}$$

To estimate L_2 error of $\|f-\tilde{f}\|^2$, we note that

$$\begin{aligned}
&\|f-\tilde{f}\|^2\\
&=\|(P\beta)^T(PF)-\left(\begin{bmatrix}I_p & 0\end{bmatrix}P\beta-V_{12}V_{22}^{-1}\begin{bmatrix}0 & I_{n-p}\end{bmatrix}P\beta\right)^T\begin{bmatrix}I_p & 0\end{bmatrix}PF\|^2\\
&=\|\left(\begin{bmatrix}0 & I_{n-p}\end{bmatrix}P\beta\right)^T\left(\begin{bmatrix}0 & I_{n-p}\end{bmatrix}PF\right)+\left(\begin{bmatrix}0 & I_{n-p}\end{bmatrix}P\beta\right)^TV_{22}^{-T}V_{12}^T\begin{bmatrix}I_p & 0\end{bmatrix}PF\|^2\\
&=\|\left(\begin{bmatrix}0 & I_{n-p}\end{bmatrix}P\beta\right)^TV_{22}^{-T}\left(V_{22}^T\begin{bmatrix}0 & I_{n-p}\end{bmatrix}PF+V_{12}^T\begin{bmatrix}I_p & 0\end{bmatrix}PF\right)\|^2\\
&=\|\left(\begin{bmatrix}0 & I_{n-p}\end{bmatrix}P\beta\right)^TV_{22}^{-T}\left(PQ\begin{bmatrix}0 & I_{n-p}\end{bmatrix}^T\right)^TPF\|^2\\
&=\|\left(\begin{bmatrix}0 & I_{n-p}\end{bmatrix}P\beta\right)^TV_{22}^{-T}\begin{bmatrix}0 & I_{n-p}\end{bmatrix}Q^TF\|^2\\
&\le\|\beta\|^2\|V_{22}^{-T}\|^2\left(\lambda_{p+1}+\cdots+\lambda_n\right)\\
&\le C\|V_{22}^{-T}\|^2(n-p)\epsilon.
\end{aligned}$$

It remains to control the term $\|V_{22}^{-T}\|^2$. By Lemma 3.1, for any orthogonal matrix Q, there exists a permutation P, such that $\|V_{22}^{-T}\|^2\le(p+1)(n-p)$. ∎

It is worth noting that the bound established in the theorem is quite loose. The final inequality in the proof straightforwardly bounds $\lambda_{p+1}+\cdots+\lambda_n$ by $(n-p)\epsilon$ despite the potential for rapid decay of the eigenvalues in practice. Moreover, the bound in Lemma 3.1 is not sharp, except for $p=1$ and $p=n-1$. We conjecture that $\frac{1}{\sqrt{n}}$ provides the sharp bound for all p, supported by extensive numerical experiments that have yet to produced a counterexample.

We then apply the theorem to the neural networks. The L-layer neural network is represented as follows:

$$\begin{aligned}
\mathcal{F}_n&=\left\{\sum_{i=1}^n\beta_j\phi_j(x;\theta)\middle|\phi_j(x;\theta)\in\mathcal{H}_L,\beta_j\in\mathbb{R},j=1,\cdots,n\right\},\\
\mathcal{H}_k&=\left\{\sigma(w_k\cdot y(x)+b_k)\middle|y(x)\in\mathbb{R}^{n_{k-1}},y_j(x)\in\mathcal{H}_{k-1},\right.\\
&\qquad\left.w_k\in\mathbb{R}^{n_{k-1}},b_k\in\mathbb{R}\right\},\quad k=1,\cdots,L,\\
\mathcal{H}_0&=\left\{x\in\Omega\subset\mathbb{R}^d\right\}.
\end{aligned}\tag{3.1}$$

where $\{\beta_j\}_{j=1}^n$ and $\theta=\{(W_k,B_k)\}_{k=1}^L$ are trainable coefficients, and σ is the activation function satisfying the universal approximation theorem of neural networks [15]. The two-layer (shallow) neural network is a special case when $L=1$.

Consider the following loss function:

$$\min_{u\in\mathcal{F}_n}\mathcal{L}(u)=\|\mathcal{G}(u)-f\|^2,\tag{3.2}$$

where $f\in L^2(\Omega)$ represents the target function or data. In this work, we always assume that the exact solution u^* does not belong to the function class $\mathcal{F}_n$. Obviously, with standard definition of the linearly independence, the Gram matrix of a optimizer $u_n^*\in\mathcal{F}_n$ should be full rank. Otherwise, some redundant neurons in the last layer can be deleted. More concisely, $r(M_{u_n^*})=n$. Hence, for a wider network we can always find a better approximation, i.e.,

$$\operatorname{dist}(u^*,\mathcal{F}_n)<\operatorname{dist}(u^*,\mathcal{F}_m),\quad m<n,$$

where $\text{dist}(u, A) = \min_{v\in A} \|u-v\|$. However, the situation will become quite different if we consider the concept of the ϵ-rank. We get a more precise relationship between the loss function and the ϵ-rank.

THEOREM 3.3. *Given the problem $\mathcal{G}(u) = f$ and $\mathcal{G}^{-1}$ being the solution operator, assume that the problem satisfies the following stability condition, for any u, v,*

$$\|\mathcal{G}^{-1}(u) - \mathcal{G}^{-1}(v)\| \le C_S\|u - v\|. \tag{3.3}$$

Let u_n be an arbitrary approximation in $\mathcal{F}_n$ with the ϵ-rank equalling to p, i.e.,

$$u_n(x;\theta) = \sum_{j=1}^{n} \beta_j \phi_j(x;\theta),$$

where $\|\beta\|^2 \le C$ and $r_\epsilon(M_{u_n}) = p$. Then

$$\sqrt{\mathcal{L}(u_n)} \ge \frac{1}{C_s}\left(dist(u^*, \mathcal{F}_p) - \sqrt{C(p+1)(n-p)^2\epsilon}\right), \tag{3.4}$$

where $u^* = \mathcal{G}^{-1}(f)$ is the exact solution, $\mathcal{L}(u) = \|\mathcal{G}(u) - f\|^2$ is the loss function.

Proof. By Theorem 3.2, there exists a $u_p \in \mathcal{F}_p$ satisfying $\|u_p - u_n\|^2 \le C(p+1)(n-p)^2\epsilon$. A triangular inequality

$$\begin{aligned}
\text{dist}(u^*, \mathcal{F}_p) &\le \|u^* - u_p\| \\
&\le \|u_p - u_n\| + \|u_n - u^*\| \\
&\le \sqrt{C(p+1)(n-p)^2\epsilon} + \|u_n - u^*\| \\
&= \sqrt{C(p+1)(n-p)^2\epsilon} + \|\mathcal{G}^{-1}(\mathcal{G}(u_n)) - \mathcal{G}^{-1}(f)\| \\
&\le \sqrt{C(p+1)(n-p)^2\epsilon} + C_S\|\mathcal{G}(u_n) - f\|
\end{aligned}$$

gives the desired inequality. ∎

Remark 3.4. The stability assumption (3.3) is reasonable for well-posed problems. For instance, in the function fitting problems, $\mathcal{G} = \mathcal{I}$, $C_S = 1$, and the loss function is $\mathcal{L}(u) = \|u - f\|^2$. In the case using PINN to solve Poisson equation $-\Delta u = f$, $C_S = \|(-\Delta)^{-1}\|$ is uniformly bounded.

Remark 3.5. When the ϵ-rank of the neural network approximator $u_n \in \mathcal{F}_n$ is $r_\epsilon(M_{u_n}) = p$, the result of the theorem can be expressed as

$$\sqrt{\mathcal{L}(u_n)} \ge \frac{\text{dist}(u^*, \mathcal{F}_p)}{C_S} - O(\sqrt{\epsilon}).$$

Recall that ϵ is a fixed hyperparameter, and it is chosen sufficiently small, which yields the loss function has a lower bound in terms of $\text{dist}(u^*, \mathcal{F}_p)$. During the training process, to minimize the loss function $\mathcal{L}(u)$, there must be a decrease in $\text{dist}(u^*, \mathcal{F}_p)$, implying an increase in the ϵ-rank of the neuron functions.

Remark 3.6. If ϵ is sufficiently small and the loss function is minimized to an optimal level while p remains small relative to n, then Theorem 3.3 offers an alternative explanation: p neurons can approximate the solution well, and more features are unnecessary. This is one of the reasons why neurons pruning [37, 25] is feasible in computation.

4. ϵ-rank in Existing Methods. This section presents how the ϵ-rank evolves in various existing methodologies. A considerable number of methods facilitate the linear independence of neuron functions during the initial stages of the training process. These approaches have been demonstrated to accelerate of the training process and enhance the accuracy of the resulting models. Based on this, it is beneficial to ensure that the neuron functions exhibit a relatively high ϵ-rank at the outset.

The neural network structure is defined as (1.1). Unless otherwise mentioned, the activation function is hyperbolic tangent function $\sigma(x) = \tanh(x)$. L^2 norm is used to measure errors between predicted solution u and exact solution or reference solution u^*.

$$e = \|u - u^*\|, \qquad \tilde{e} = \frac{\|u - u^*\|}{\|u^*\|},$$

where $\|u\| = \left(\int_\Omega u^2 \mathrm{d}x\right)^{\frac{1}{2}}$ is the short note of L^2 norm.

In numerical calculations, we extend the ϵ-linear independence to the discrete form. Given n functions discretized on m nodes

$$D = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \phi_1(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \ddots & \vdots \\ \phi_1(x_m) & \cdots & \phi_n(x_m) \end{pmatrix},$$

the mass matrix is computed by $M = D^T W D$, where W is the weight matrix. The ϵ-linear independence of these n functions is

$$r_\epsilon(M) = |A|, \quad A = \{|\lambda(M)| > \epsilon\},$$

where $|A|$ is the cardinal number of A, and the tolerance is given $\epsilon = 10^{-6}$.

The rank of mass matrix, $r_\epsilon(M)$ is employed to measure the linear independence of the basis functions.

In low-dimensional cases, M is approximated using numerical integration, while in high dimensions, M is approximated by Monte Carlo integration. For simplicity, we generate one set of integration data points $\{x\}_{k=1}^m$, which can be Gaussian integral points or uniform mesh for $d = 1, 2$. Then

$$(M)_{ij} \approx \sum_{k=1}^{m} w_k \phi_i(x_k)\phi_j(x_k), \quad \text{numerical integration} \tag{4.1}$$

where w_k are the integral weights. For example, $w_0 = w_m = \frac{1}{2}$ and $w_i = 1, i = 2, \cdots, m-1$ is the trapezoidal formulation.

The notations are listed in Table 1.

4.1. Deterministic Initialization. This subsection evaluates the feasibility of directly constructing highly linearly independent neuron functions through initialization.

Example 4.1 (function fitting with initialization, Figure 6). In this example, we consider a function fitting problem where the target function is also defined as (2.5)

$$u(x) = \cos(x) + \cos(2x) + \cos(30x), \qquad x \in [-1, 1].$$

Table 1: The List of Notations

Notation	Stands for ...
L	Depth of hidden layers
n	Width of hidden layers
N	Number of total parameters
m	Sample size
d	Dimension of the input
ϕ	The neurons of the output layer
β	Coefficients of the output layer
$M(\phi)$	The mass matrix of basis functions $\{\phi_j\}_{j=1}^n$
$r_\epsilon(M)$	The ϵ-rank of the mass matrix
ϵ	Tolerance of eigenvalues

We compare two initialization methods: deterministic and random initialization. For the deterministic initialization, the first layer of the neural network is initialized as:

$$y_1 = \sigma(w_0 x + b_0), \tag{4.2}$$

where the weights and biases are given by $(w_0)_j = \frac{n}{2},\ (b_0)_j = \frac{n}{2} - j,\ j = 1, \cdots, n.$

As discussed in section 2, a set of linearly independent neuron functions is important for neural network performance. A straightforward method to improve the ϵ-rank of neurons is to expand the range of the uniform distribution. For example, set $\theta \sim U(-m, m)$. While this approach can enhance the linear independence of the neuron basis functions, it is not commonly used in practice due to the potential issues it introduces. As mentioned in [9], such an initialization strategy can lead to problems like exploding gradients and the saturation of activation functions, which negatively affect the training process.

Inspired by traditional numerical methods, we propose an alternative initialization method for the first layer of the neural network when $d = 1$, which guarantees a sufficiently high ϵ-rank of neuron functions. Specifically, the initialization for the first hidden layer is given as (4.2), which can be rewritten in the form of

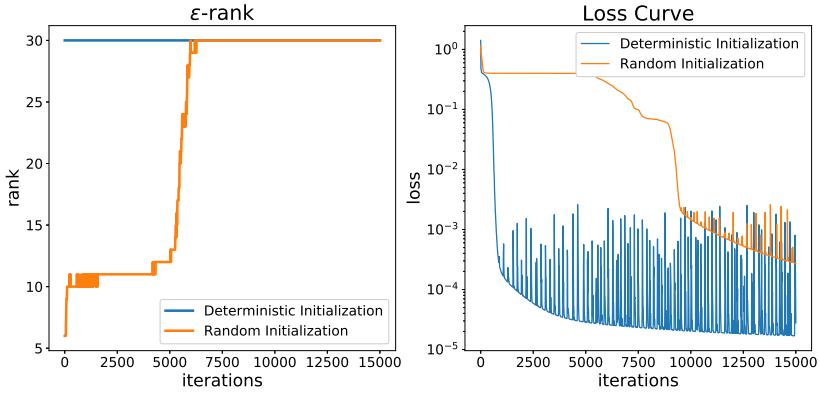
$$(y_1)_j = \tanh(\frac{n}{2}(x - x_j)), \quad j = 1, \cdots, n, \tag{4.3}$$

初始值如此y所示，实现了高epsilon-rank

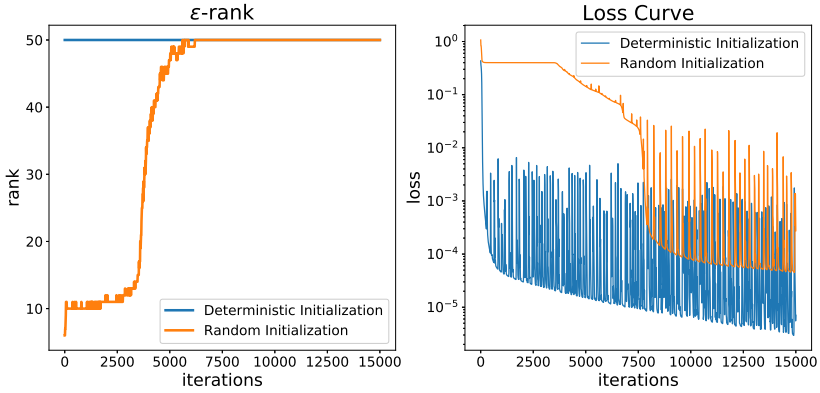
where $\{x_j\}$ are the uniform grids on $[-1, 1]$.

The result is shown in Figure 6. This figure clearly illustrates that once the neuron functions exhibit sufficient linear independence, the loss function will be rapidly optimized. Comparing the blue curve (loss $= 10^{-5}$) in Figure 6(a) and the orange curve (loss $= 10^{-4}$) in Figure 6(b), even a shallow and narrow network performs better than a deeper and wider network. This indicates that a good initialization method can greatly reduce training time and improve training accuracy.

We now demonstrate that the staircase phenomenon also arises when solving partial differential equations and that appropriate initialization techniques can significantly enhance performance. Since the work of [29], physics-informed neural networks (PINN) have achieved remarkable success in the field of computational science. Numerous studies have highlighted the advantages of neural network based algorithms



(a) $L = 2, n = 30$



(b) $L = 4, n = 50$

Fig. 6: (Example 4.1) The training process of neural networks with default settings and deterministic initialization. The left figure shows the linear independence and the right figure plots the losses. The blue curve gives the result of deterministic initialization method and the orange curve is under default settings.

for partial differential equations in high dimensional problems [31, 8, 7], irregular domains [38, 33] and systems with complex mechanisms [22, 20].

Consider the following boundary value problem:

Example 4.2 (Solving Possion's equation by PINN, Figure 7).

$$\begin{cases} -\Delta u = f, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases}$$

where $\Omega = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]^2$, $f(x, y) = 8 \sin 2x \sin 2y$. The analytic solution of this differential equation is $u(x, y) = \sin 2x \sin 2y$. The loss function is defined as

$$\mathcal{L}(u) = \|\Delta u + f\|^2_{L^2(\Omega)} + \|u\|^2_{L^2(\partial\Omega)}. \tag{4.4}$$

The previously used initialization method becomes ineffective in this context, as the problem is two-dimensional. In [41], the authors proposed a method for generating uniformly distributed neurons in shallow networks to ensure equal expressive power across all regions of the domain. Inspired by this approach, we can separate the neuron functions in the first hidden layer by employing a uniformly distributed initialization method. This method re-parameterizes the weight and bias of each neuron in the first hidden layer as:

$$\sigma(w_j^T x + (b_0)_j) = \sigma(\gamma_j(a_j^T x + r_j)), \tag{4.5}$$

where w_j^T is the j-th rows of W_0, $\|a_j\|^2 = 1$, and $\gamma_j \geq 0$. $\{a_j\}_{j=1}^n$ are initialized by uniformly sampled on the unit sphere, and $\{r_j\}_{j=1}^n$ are initialized by uniformly sampled on a bounded positive real line related to Ω. This initialization method generate neurons with uniform hyperplane density in the region, termed as uniform density initialization (UDI).

The result is presented in Figure 7, which clearly demonstrate that solving PDEs with neural networks also exhibits the staircase phenomenon. Furthermore, the figure highlights that an effective initialization technique, which ensures a high ϵ-rank, can significantly accelerate the training process.

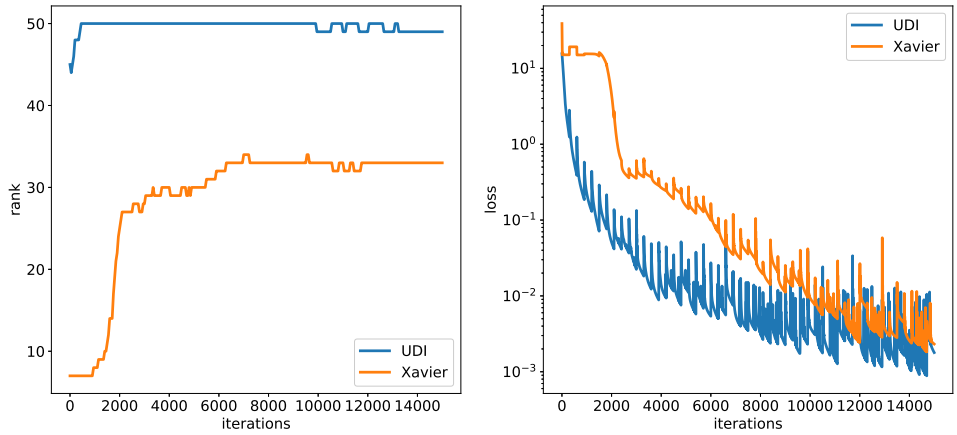


Fig. 7: (Example 4.2) The training process of solving 2-D Poisson's equation with default Xavier initialization and uniformly density initialization (UDI). The left figure shows the ϵ-rank and the right figure shows losses. The blue curve gives the result of uniformly density initialization and the red curve is under Xavier initialization.

4.2. Partial of Unity. Next, we evaluate the ϵ-rank in two methods that do not involve training process. Both extreme learning machine (ELM) [17] and random features [28] are widely used techniques in deep learning. Notably, neither method involves training the hidden layers, offering computationally efficient solutions. In these approaches, the linear independence of the neuron functions depends solely on the initialization and network structure employed.

An important improvement is the partition of unity method (PoU) technique, employed in the random feature method (RFM) [2] and local extreme learning machine [6]. In RFM, the approximate solution is expressed as a linear combination of random

features combined with the PoU, as follows:

$$u_R(x) = \sum_{i=1}^{m} \psi_i(x) \sum_{j=1}^{J_R} u_{ij}\phi_{ij}(x),$$

where u_{ij} are unknown coefficients. In RFM, N points $\{x_i\}_{i=1}^N$ are chosen from Ω, typically uniformly distributed. Then Ω is decomposed to N disjoint subdomains $\{\Omega_i\}_{i=1}^N$ with $x_i \in \Omega_i$. For each Ω_i, a PoU function ψ_i is constructed with support Ω_i, i.e., $\mathrm{supp}(\psi_i) = \Omega_i$. The commonly used PoU function is

$$\psi_i(x) = \mathbb{I}_{\Omega_i}(x).$$

Since $|\Omega_i \cap \Omega_j| = 0$, it is clear that $\{\psi_i \sum_{j=1}^{J_R} u_{ij}\phi_{ij}\}_{i=1}^N$ are ϵ-linearly independent when $\|\psi_i \sum_{j=1}^{J_R} u_{ij}\phi_{ij}\|^2 > \epsilon$. The extreme learning machine is modeled as

$$u_E(x) = \sum_{j=1}^{J_E} u_j \phi_j(x),$$

which can be seen as a random feature method with no subdomains ($N = 1$).

Example 4.3 (Two dimensional function fitting, Figure 8).

$$u^*(x, y) = \cos x \cos y + \cos 10x \cos 10y, \quad \Omega = [-1, 1]^2.$$

In this example, we compare the performance of modeling with and without the partition of unity (PoU) technique, regarded as the RFM and ELM respectively. The number of neurons is set to $n = 900$. In the RFM, these 900 neurons are divided into 3×3 sub-intervals, i.e., $i = 9, J_R = 100$ and in the ELM, $J_E = 900$. The coefficients of the output layer in both methods are determined using the least squares method.

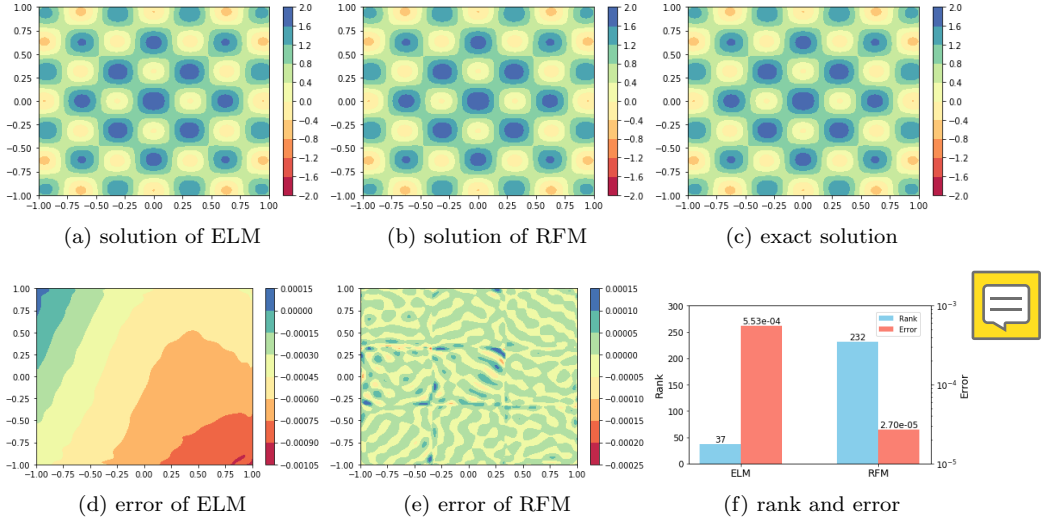
The results, presented in Figure 8, demonstrate that, for the same number of neurons, the RFM achieves greater linear independence due to the compact support in each subdomain, resulting in higher accuracy. This example clearly illustrates that, under identical configurations, achieving a high linear independence through specific techniques can significantly enhance network performance. Furthermore, an approximate inverse proportional relationship between the ϵ-rank and the error is observed.

4.3. ResNet. The network structure has a significant impact on the linear independence of neuron functions. The Residual Network (ResNet), is a deep neural network architecture introduced by [12]. ResNet has been widely adopted in the field of deep learning, particularly for the training of very deep networks.

The typical structure of the ResNet is the residual block, which is given as

$$y = R(x) = x + \sigma(W_1\sigma(W_2x + b_2) + b_1).$$

The shortcut connection allows the input of the block to bypass the transformations and connect directly to its output. This shortcut provides a direct path for the gradient, facilitating more effective backpropagation and addressing the degradation problem, where deeper networks may perform worse than their shallower counterparts.



(a) solution of ELM (b) solution of RFM (c) exact solution

(d) error of ELM (e) error of RFM (f) rank and error

Fig. 8: (Example 4.3) (a)-(e) are the solutions and point-wise errors of random feature method and extreme learning machine method. The last subfigure (f) shows the rank and L^2 error of two methods.

When the neuron functions in deeper layers are connected to the neuron functions in shallower layers via a residual block, i.e.,

$$\phi^{L+1}(x) = R(\phi^L(x)),$$

this architecture demonstrates an additional advantage: it helps sustain the growth of the ϵ-rank.

Example 4.4 (Function fitting with ResNet, Figures 9 and 10). This example examines the ϵ-rank in each layer of the ResNet compared to MLP. To ensure a comparable number of parameters and to make each layer of neuron functions observable, we employ a one-layer residual block. The ResNet structure is defined as follows:

$$\begin{cases} y_0 = x, \\ y_{k+1} = y_k + \sigma(W_k y_k + b_k), \quad k = 0, \cdots, L-1, \\ y = \beta \cdot y_L. \end{cases} \tag{4.6}$$

The problem is the same function fitting problem as Example 2.6.

In Figure 9, we observe that the ϵ-rank of ResNet is initially higher and grows faster to full rank compared to the MLP. Furthermore, as shown in Figure 10, the residual block structure ensures greater rank growth in the deeper neuron functions. This example demonstrates that the linear independence of neuron functions varies across different network structures. Selecting a network architecture that promotes linear independence among neuron functions enables the capture of more features, thereby improving performance.

5. Concluding Remarks. In summary, this research provides a novel perspective on the training dynamics of deep neural networks by drawing connections to

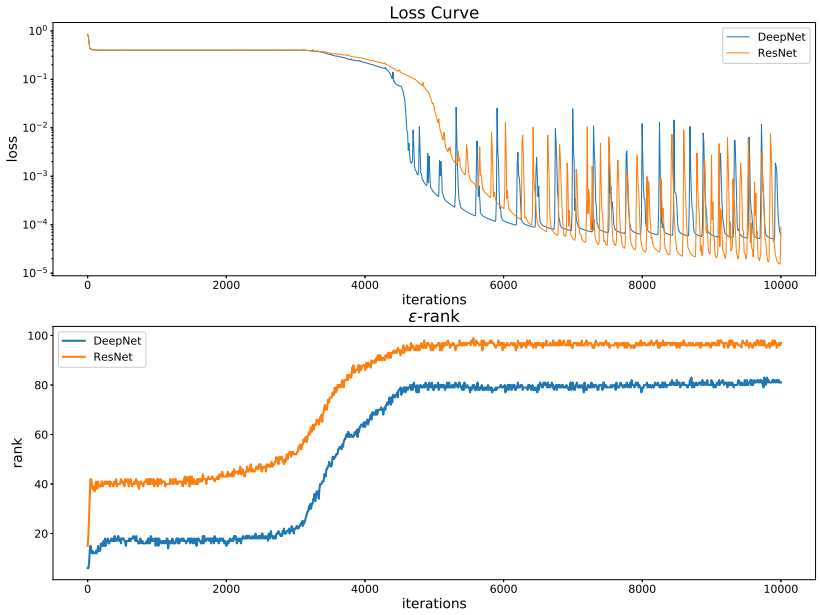


Fig. 9: (Example 4.4) The training result between ResNet and fully connected neural network.

traditional numerical analysis. A key finding of our study is the identification of the *staircase phenomenon*, which describes a stepwise increase in the linear independence of neuron functions during the training process, typically associated with rapid decreases in the loss function. This finding highlights the importance of establishing a diverse and robust set of neuron functions in the early stages of training, which promotes both efficiency and convergence in model optimization.

Theoretical analyses and numerical experiments confirm that a set of linearly independent basis neuron functions is essential for effectively minimizing the loss function of neural networks. The training process can be significantly accelerated by leveraging appropriate techniques, such as deterministic initialization methods, constructing efficient network architectures, and employing proper domain partitioning. These strategies have shown effective in promoting the linear independence of neuron functions, providing insight into the success of certain neural network approaches.

This study offers a deeper understanding of the mechanisms underlying deep learning by bridging neural network framework with traditional numerical methods. It provides a solid foundation for future innovations in network initialization, training methodologies, and the design of more efficient models.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (NSFC) Grant No. 12271240, and the Shenzhen Natural Science Fund (No. RCJC20210609103819018).

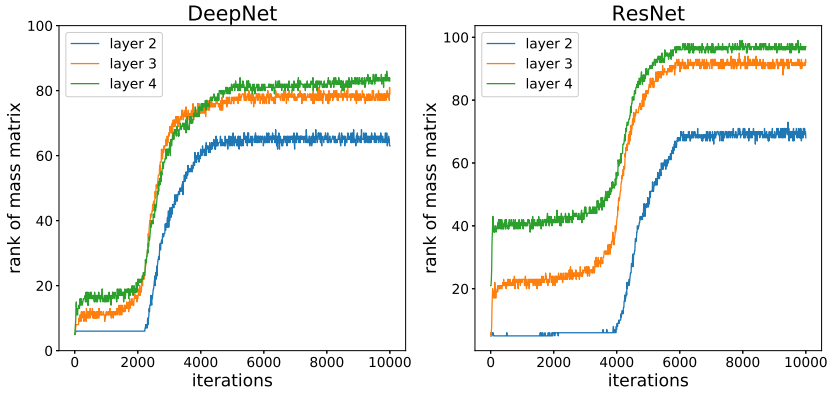


Fig. 10: (Example 4.4) The ϵ-rank for each layer of neuron functions in DeepNet and ResNet.

REFERENCES

[1] A. Bietti and J. Mairal, *On the Inductive Bias of Neural Tangent Kernels*, in Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 12893–12904.

[2] J. Chen, W. E, and Y. Luo, *The Random Feature Method for Time-Dependent Problems*, East Asian Journal on Applied Mathematics, 13 (2023), pp. 435–463.

[3] J. Chen, W. E, and Y. Sun, *Optimization of Random Feature Method in the High-Precision Regime*, Communications on Applied Mathematics and Computation, 6 (2024), pp. 1490–1517.

[4] Z. Chen, Y. Cao, Q. Gu, and T. Zhang, *A Generalized Neural Tangent Kernel Analysis for Two-layer Neural Networks*, in Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., 2020, pp. 13363–13373.

[5] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, *Sharp Minima Can Generalize For Deep Nets*, in Proceedings of the 34th International Conference on Machine Learning, PMLR, July 2017, pp. 1019–1028.

[6] S. Dong and Z. Li, *Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations*, Computer Methods in Applied Mechanics and Engineering, 387 (2021), p. 114129.

[7] W. E, J. Han, and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Communications in Mathematics and Statistics, 5 (2017), pp. 349–380.

[8] W. E and B. Yu, *The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.

[9] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[10] J. He, L. Li, and J. Xu, *ReLU deep neural networks from the hierarchical basis perspective*, Computers & Mathematics with Applications, 120 (2022), pp. 105–114.

[11] J. He, L. Li, J. Xu, and C. Zheng, *Relu Deep Neural Networks and Linear Finite Elements*, Journal of Computational Mathematics, 38 (2020), pp. 502–527.

[12] 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, 2016, pp. 770–778.

[13] S. Hochreiter and J. Schmidhuber, *Flat Minima*, Neural Computation, 9 (1997), pp. 1–42.

[14] Y. P. Hong and C.-T. Pan, *Rank-revealing QR factorizations and the singular value decomposition*, Mathematics of Computation, 58 (1992), pp. 213–232.

[15] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks,

4 (1991), pp. 251–257.
[16] G.-B. Huang, L. Chen, and C.-K. Siew, *Universal Approximation using Incremental Constructive Feedforward Networks with Random Hidden Nodes*, IEEE Transactions on Neural Networks, 17 (2006), pp. 879–892.
[17] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, *Extreme learning machine: Theory and applications*, Neurocomputing, 70 (2006), pp. 489–501.
[18] A. Jacot, F. Gabriel, and C. Hongler, *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*, in Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018, pp. 8580–8589.
[19] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, in International Conference on Learning Representations, 2017.
[20] C. L. Wight and J. Zhao, *Solving Allen-Cahn and Cahn-Hilliard Equations Using the Adaptive Physics Informed Neural Networks*, Communications in Computational Physics, 29 (2021), pp. 930–954.
[21] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, *Visualizing the Loss Landscape of Neural Nets*, in Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018, pp. 6391–6401.
[22] Z. Liu, W. Cai, and Z.-Q. John Xu, *Multi-Scale Deep Neural Network (MscaleDNN) for Solving Poisson-Boltzmann Equation in Complex Domains*, Communications in Computational Physics, 28 (2020), pp. 1970–2001.
[23] R. E. Meethal, A. Kodakkal, M. Khalil, A. Ghantasala, B. Obst, K.-U. Bletzinger, and R. Wüchner, *Finite element method-enhanced neural network for forward and inverse problems*, Advanced Modeling and Simulation in Engineering Sciences, 10 (2023), p. 6.
[24] S. K. Mitusch, S. W. Funke, and M. Kuchta, *Hybrid FEM-NN models: Combining artificial neural networks with the finite element method*, Journal of Computational Physics, 446 (2021), p. 110651.
[25] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, *Importance Estimation for Neural Network Pruning*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 11264–11272.
[26] N. H. Nelsen and A. M. Stuart, *The Random Feature Model for Input-Output Maps between Banach Spaces*, SIAM Journal on Scientific Computing, 43 (2021), pp. A3212–A3243.
[27] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, *On the Spectral Bias of Neural Networks*, in Proceedings of the 36th International Conference on Machine Learning, PMLR, May 2019, pp. 5301–5310.
[28] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, Advances in neural information processing systems, 20 (2007), pp. 1177–1184.
[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019), pp. 686–707.
[30] P. Ramuhalli, L. Udpa, and S. Udpa, *Finite-Element Neural Networks for Solving Differential Equations*, IEEE Transactions on Neural Networks, 16 (2005), pp. 1381–1392.
[31] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339–1364.
[32] S. Wang, X. Yu, and P. Perdikaris, *When and why PINNs fail to train: A neural tangent kernel perspective*, Journal of Computational Physics, 449 (2022), p. 110768.
[33] W. Wang, BoZhang and W. Cai, *Multi-scale deep neural network (MscaleDNN) methods for oscillatory stokes flows in complex domains*, Communications in Computational Physics, 28 (2020), pp. 2139–2157.
[34] Z.-Q. J. Xu, *Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks*, Communications in Computational Physics, 28 (2020), pp. 1746–1767.
[35] Z.-Q. J. Xu, Y. Zhang, and T. Luo, *Overview Frequency Principle/Spectral Bias in Deep Learning*, Communications on Applied Mathematics and Computation, (2024).
[36] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, *Training Behavior of Deep Neural Network in Frequency Domain*, in Neural Information Processing, T. Gedeon, K. W. Wong, and M. Lee, eds., Cham, 2019, Springer International Publishing, pp. 264–274.
[37] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, *NISP: Pruning Networks Using Neuron Importance Score Propagation*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9194–9203.
[38] Y. Zang, G. Bao, X. Ye, and H. Zhou, *Weak adversarial networks for high-dimensional partial differential equations*, Journal of Computational Physics, 411 (2020), p. 109409.

[39] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*, in Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds., Cham, 2014, Springer International Publishing, pp. 818–833.
[40] S. Zhang, H. Zhao, Y. Zhong, and H. Zhou, *Why shallow networks struggle with approximating and learning high frequency: A numerical study*, 2023, https://arxiv.org/abs/2306.17301.
[41] Z. Zhang, F. Bao, L. Ju, and G. Zhang, *Transferable Neural Networks for Partial Differential Equations*, Journal of Scientific Computing, 99 (2024), p. 2.