Roughness Index for Loss Landscapes of Neural Network Models of Partial Differential Equations*

Keke Wu  
School of Mathematical Sciences  
Shanghai Jiao Tong University  
Shanghai, China  
wukekever@sjtu.edu.cn

Jingrun Chen  
School of Mathematical Sciences  
and Suzhou Institute for Advanced Research  
University of Science and Technology of China  
Hefei, China  
jingrunchen@ustc.edu.cn

Xiangru Jian  
School of Data Science  
City University of Hong Kong  
Hong Kong, China  
xiangjian2-c@my.cityu.edu.hk

Xiang ZHOU  
School of Data Science  
and Department of Mathematics  
City University of Hong Kong  
Hong Kong, China  
xizhou@cityu.edu.hk

Rui Du  
School of Mathematical Sciences  
Soochow University  
Suzhou, China  
durui@suda.edu.cn

Abstract—Loss landscape is a useful tool for characterizing and comparing neural network models. The main challenge for analysis of loss landscape for the deep neural networks is that they are generally highly nonconvex in very high-dimensional space. In this paper, we develop the “roughness” concept for understanding such landscapes in high dimensions and apply this technique to study two neural network models arising from solving differential equations. Our main innovation is the proposal of a well-defined and easy-to-compute roughness index (RI) which is based on the mean and variance of the (normalized) total variation for one-dimensional functions projected on randomly sampled directions. A large RI at the local minimizer indicates an oscillatory landscape profile and indicates a severe challenge for the first-order optimization method. Particularly, we observe the increasing-then-decreasing pattern for RI along the gradient descent path in most models. We apply our method to two types of loss functions used to solve partial differential equations (PDEs) when the solution of PDE is parametrized by neural networks. Our empirical results on these PDE problems reveal important and consistent observations that the landscapes from the deep Galerkin method around its local minimizers are less rough than the deep Ritz method.

Index Terms—roughness index, landscapes, total variation

I. INTRODUCTION

In recent years, solving partial differential equations (PDE) by deep neural networks (DNNs) has brought significant interests from the community of scientific computing; see [1] for reviews and references therein. Due to its powerful representation ability, a DNN can well approximate a target function in high dimensions. Given a PDE, the basic idea is to use a DNN as the trial function to approximate the PDE solution. The optimal set of parameters in the DNN is obtained by minimizing a loss function in different forms [2]–[4]. Since the loss function lives in the high-dimensional parameter space and is highly non-convex, it is difficult to find the global minimizer. The minimization problem is often solved by the stochastic gradient descent (SGD) method [5]. The complexity of loss landscapes makes the training process and the numerical results highly dependent on the DNN structure, the optimization method, and the initialization [6].

Efforts to better understand loss landscapes theoretically have included studies on specialized neural networks [7], [8], the geometry of local minima [9]–[11], energy barriers [12], and the studies for mean field limit [13] or neural tangent kernel limit [14]. For a general DNN, visualization is also a very intuitive tool in practice. However, visualizing the loss function of a general Deep Neural Network (DNN) can be challenging due to the high dimensionality of the parameter space. One approach is to project the loss function onto a low-dimensional space, using random direction selection and filter-wise normalization [15]. This method has been employed to demonstrate the superiority of certain residual neural networks over fully connected neural networks [16]. Furthermore, the volume of basins of attraction (local minimizers) has been explored to characterize the flatness of minima [17]. While this concept is compelling, it proves difficult to apply in practice.

We are interested in understanding and comparing two loss functions within the context of solving Partial Differential Equations (PDEs). In this setting, the same network architectures and training data are used to solve the identical PDEs, with the only difference being the form of the loss functions. As different loss functions are applied to the same PDE, with the only difference being the form of the loss functions. The work of Chen is partially supported by National Key R&D Program of China (No. 2022YFA1005200 and No. 2022YFA1005203), NSFC Major Research Plan - Interpretable and General-purpose Artificial Intelligence (No. 92270001 and No. 92270205), Anhui Center for Applied Mathematics, and the Major Project of Science & Technology of Anhui Province (No. 202203a05020050). This work of Du is partially supported by National Natural Science Foundation of China via grant 12271360. The work of Zhou is partially supported by Hong Kong RGC GRF 11307319, 11308121, 11318522, and the NSFC/RGC Joint Research Scheme [RGC Project No. N_CityU102/20 and NSFC Project No. 12061160062].
examine two representative methods for solving PDEs with Deep Neural Networks (DNN): the variation-based model, known as the Deep Ritz Method (DRM) [2], and the residual-based model, the Deep Galerkin Method (DGM) [3].

As widely recognized [18]–[20], the loss function is complex due to non-convexity, characterized by many local minimizers with the valley of a “good” minimizer. Unlike traditional gradient descent which is easily trapped in one of many neighboring local minimizers, the noise in the SGD can in general overcome many small barriers connecting adjacent local minimizers in that oscillatory neighboring region and eventually reach that “good” minimizer. These “good” minimizers are theorized to possess “wide” and “flat” geometric properties, consequently offering better generalization capabilities. The training process relies on noise introduced by the stochastic optimization method to surmount these minor barriers, thereby achieving improved accuracy and generalization error. As such, the loss landscape is fundamentally “rough”, with the training process representing an exploration of this rough landscape before ultimately arriving at the final solution. It’s worth noting that the landscape here refers to the population risk, and the roughness is not a result of the randomness of training data in empirical risk.

In this paper, we introduce a quantitative index, referred to as the “roughness index” (RI), to encapsulate the concept of roughness. This index measures the degree of roughness at any given point in the landscape for different models solving the PDEs. It characterizes the cumulative effect of the small-scale oscillatory form of the loss function within the vicinity of numerically obtained minimizers. The roughness index can be assigned to each minimizer identified by the standard stochastic optimization approach. Importantly, this index is delocalized, meaning it does not derive from the minimizer’s eigenvalues and transcends the infinitesimal quadratic approximation used in previous works. The index may vary depending on the size of the neighborhood, which in our computations is represented by a box. Ideally, this length scale should correspond to the typical size scale of the basin of attraction.

In practice, we compute the index using varied box sizes and identify consistent results within a range of appropriate sizes.

By calculating the Roughness Index (RI) for various local minimizers of the DGM and the DRM applied to the Poisson equation, we discern consistent and distinctive differences: DGM’s minimizers exhibit a smaller RI, while DRM’s minimizers demonstrate a larger RI. This could suggest that DRM might be easier to train than DRM.

We also monitor the RI value along the Stochastic Gradient Descent (SGD) training trajectory and observe that for typical initialized parameters of the Neural Network (NN), the roughness index is small. Then, DRM’s roughness index gradually increases as the training approaches the minimizers, indicating that the optimization path transitions from a relatively smooth and monotonic region to an increasingly rough region where the numerically obtained minimizer resides.

In essence, by examining the roughness index in the high-dimensional parameter space, we can uncover several intriguing and phenomenal insights about the loss landscape in a quantitative manner that has yet to be explored. This roughness index is not confined to NN models for PDEs, but could potentially serve as a tool for analyzing general machine-learning landscapes.

This paper is organized as follows. We first give an introduction of methods for solving PDEs by DNNs: the DRM and DGM. Section II is our main part to define and compute roughness index. Section III applies the RI to different models, different neural networks, and different dimensions and different PDEs. Conclusive remarks are drawn in Section IV.

II. RELATED WORKS

A. Solving PDEs by deep neural networks

When using a NN to solve a given PDE, there are multiple choices to construct the loss function. If the PDE is derived from a variational problem, then this variational functional can be defined as the loss function; see DRM [2] for example. In contrast, DGM [3] uses the mean-square error of the PDE residual. For completeness, we shall first review these two methods for the elliptic equation where the variational loss residual blocks. For the

\[ L^i[x] = L^i[x^i] + \sigma(W_{i} \cdot \sigma(W_{i} \cdot L^i[x] + b_i) + b_i^2), \]

where \( i = 0, \ldots, N - 1 \). The input and the output layers are

\[ L^0(x) = L^0 = W^0 \cdot x + b^0 \quad \text{and} \quad L^{N+1}[x] = L^{N+1} \cdot x + b^{N+1} \]

with \( W^0 \in \mathbb{R}^{w \times d}, b^0 \in \mathbb{R}^{w \times 1} \) and \( W^{N+1} = W^{N+1} = W^{N+1} = W^{N+1} \in \mathbb{R}^{w \times 1} \). For the i-layer of FCNet with totally \( 2N + 2 \) layers, let \( L^i[x] \in \mathbb{R}^{w \times 1} \) be the input, \( W^i \in \mathbb{R}^{w \times w} \) and \( b^i \in \mathbb{R} \) be the weight matrices and bias vectors, \( \sigma(\cdot) \) the activation function, then the output \( L^{i+1}[x] \) can be written as

\[ L^{i+1}[x] = L^i[x] + \sigma(W^i \cdot \sigma(W^i \cdot L^i[x] + b_i^1) + b_i^2), \]

where \( i = 0, \ldots, N - 1 \). The input and the output layers are

\[ L^0(x) = W^0 \cdot x + b^0 \quad \text{and} \quad L^{N+1}[x] = L^{N+1} \cdot x + b^{N+1} \]

with \( W^0 \in \mathbb{R}^{w \times d}, b^0 \in \mathbb{R}^{w \times 1} \) and \( W^{N+1} = W^{N+1} = W^{N+1} = W^{N+1} \in \mathbb{R}^{w \times 1} \). For the i-layer of FCNet with totally \( 2N + 2 \) layers, let \( L^i[x] \in \mathbb{R}^{w \times 1} \) be the input, \( W^i \in \mathbb{R}^{w \times w} \) and \( b^i \in \mathbb{R} \) be the weight matrices and bias vectors, then the output \( L^{i+1}[x] \) can be written as

\[ L^{i+1}[x] = \sigma(W^i \cdot L^i[x] + b_i^1), \]

where \( i = 0, \ldots, N - 1 \). The input and the output layers are

\[ L^0(x) = W^0 \cdot x + b^0 \quad \text{and} \quad L^{N+1}[x] = L^{N+1} \cdot x + b^{N+1} \]

The number of neurons in each hidden layer (neural width) is \( w \). Therefore, the total number of parameters in ResNet or FCNet is \( 2Nw^2 + (d + 2N + 2)w + 1 \). We use the smooth activation function, \( \text{swish} \) function \( (x(1 + e^{-x})^{-1}) \). Boundary condition is enforced exactly by constructing a special form of the solution embedded with neural network. DGM and DRM only differ by their loss functions in terms of \( \theta \), which are

\[ J_G(\theta) = \int_{\Omega} | - \Delta u(x; \theta) - f(x) |^2 dx, \]
and
\[ J_R(\theta) = \int_{\Omega} \left( \frac{1}{2} |\nabla u(x; \theta)|^2 - f(x)u(x; \theta) \right) \, dx. \tag{3} \]

A minimizer is obtained by Adam optimizer. Derivatives of \( u(x; \theta) \) are calculated by the automatic differentiation. Monte Carlo method is applied to approximate the integrals in DGM and DRM by \( N \) samples. In 1D, instead, the Simpson’s rule is used for better accuracy. One epoch refers to the period of processing \( N \) samples, i.e., one step in Adam. We typically set the batch size \( N = 200, 1000, 10000 \) in solving the 1D, 3D, 10D PDE, respectively.

### B. Eigenvalue-based index

The loss landscape is complicated and typically there are many minima of interest. For example, for a simple NN, the minima of loss function may lie in a very flat basin [8]. To understand the loss landscapes of DGM and DRM, we first consider the concept of “volume of basin of attractor” proposed in [17]. Their use of the (Lebesque) measure of the basin for each attractor is an appealing idea. However, it is almost impossible in reality to find the exact basin and precisely measure its volume in high dimensional space. As a compromise, [17] in fact used the Hessian matrix at the minimum point to represent the “volume of the basin” of this minimum point. Precisely, for a given minimizer \( \theta^* \), one can compute the Hessian \( H \) of loss function with respect to \( \theta \) and evaluate it at \( \theta^* \). Since the volume of the sublevel set of a quadratic form is proportional to the product of eigenvalues, [17] used the logarithm of the product of top-\( k \) eigenvalues (\( k \) is truncated to keep only significant nonzero eigenvalues) of \( H(\theta^*) \) to approximate the inverse volume of basin of attractor
\[ V(k) := \sum_{i=1}^{k} \log_{10}(\lambda_i(H(\theta^*))). \tag{4} \]

(4) provides a quantitative characterization of the size of the basin around a minimizer for the local quadratic approximation of the landscape. A small \( V \) means a “flat” valley near \( \theta^* \) and is regarded to have a large volume of basin, which arguably is able to generalize well [15], [17]. We emphasize that the index \( V \) in (4) only relies on the Hessian information at the minimizer, thus it is essentially a local quantity for characterizing the flatness which should not be too localized. The assumption behind this argument that the landscape around \( \theta^* \) is convex and smooth is questionable since the neighboring region for such assumptions to be valid could be very small. Therefore, it is hard to justify the applicability of this index to represent the real non-convex behaviors around the local minimum points.

### C. Normalized total variation for 1D functions

Total variation (TV) is a commonly used norm in applied mathematics for regularity of a function. For instance, TV has been used in image denoising as a penalty to suppress the spurious detail [21], [22]. It is also adopted in the statistical learning for the purpose of smoothing and regularization in fitting data. It is one of natural candidates to describe the “regularity” or “roughness” of the signals. We propose to utilize the concept of TV to construct roughness index.

Recall that the TV of a continuous function \( f \) from \([a, b]\) to \( \mathbb{R} \) is given by
\[ \text{TV}(f) = \sup_{n-1} \sum_{k=0}^{n-1} |f(x_{k+1}) - f(x_k)| \]
where the sup is taken over all possible partitions, \( a = x_0 < \ldots < x_n = b \). If \( f \) is absolutely continuous, we can write
\[ \text{TV}(f) = \int_a^b |f'(x)| \, dx. \]

The definition of TV is free of the deformation in the input variable: let \( \varphi : [a', b'] \to [a, b] \) be a diffeomorphism, then \( \text{TV}(f \circ \varphi) = \text{TV}(f) \). For two functions defined on the same domain and have the similar size of the range, the TV norm can effectively describe the heuristic concept of “roughness”. Refer to Figure 1 where the right-side function has a much larger TV. If \( f \) is monotonic, then \( \text{TV}(f) = \max f - \min f \).

![Fig. 1. Two functions with the same global minimizer but different total variations. Left: the convex function \( f(x) = -\cos(x) \) defined over \((-3, 3)\); Right: the same function added with a few high-frequency cosine modes, whose TV is the total sum of heights differences between adjacent minimizers and maximizers.](image)

There is another important interpretation for the difference in the two functions in Figure 1 from the viewpoint of SGD [23], [24]. If one applies the SGD to minimize these two functions, it takes much more time on the “more rough” function to reach the (global) optimal solution near \( x = 0 \); the momentum acceleration like Adam can mildly mitigate this slow convergence but generally speaking, the function with a larger TV is indeed harder to train. Of course, the full gradient method without noise injection fails to obtain the global minimum for the non-convex function in this case. The above interpretation of using the TV to describe the impact to the stochastic training method can be explained more precisely from the perspective of the the well-known Freidlin-Wentzell large deviation theory [25]–[27] for
\[ dX_t = -\nabla f(X_t) + \sqrt{2\epsilon}dW_t. \]

In this theory, the probability for the trajectories \( X_t \) between two given endpoints are approximately (up to the exponential
scale) determined by the so-called quasi-potential function, for small $\epsilon$. We refer to the global minimum point in Figure 1 as $o$. Then the quasi-potential $Q(o \rightarrow a)$ for transition starting from the lowest point $o$ and exiting the domain through the endpoint $a$, is the sum of all energy barriers. Therefore we have $\sum_{i=\text{enc}} Q(o \rightarrow i) + Q(i \rightarrow o) = TV(f)$ holds exactly for any 1D function defined over $[a, b]$. In this sense, $TV(f)$ represents how difficult the stochastic gradient descent approaches the lower point $o$ from one boundary of the domain and then exits the domain via either of boundary points. The bound of $TV$ is also closely relevant to the magnitudes of the Fourier coefficients. It is well known that a large Fourier coefficient at high frequency implies the function in space is more “oscillatory”. If $f$ on $[-\pi, \pi]$ has a bounded TV, then its Fourier coefficients $\hat{f}_k$ decay at least $O(1/k)$: specifically we have [28]:

$$|\hat{f}_k| \leq \frac{2}{k\pi} TV(f).$$

A small $TV(f)$ corresponds to small Fourier coefficients.

It is easy to see that $TV(\alpha f) = \alpha TV(f)$, $\alpha > 0$. But to minimize $f$ and $\alpha f$ is exactly the same computational tasks if the learning rate is rescaled accordingly. So, the index for the function should be free of such dilution operation, and as a result we propose the following modified TV

$$T(f) := \frac{1}{b-a} \frac{1}{[f]} TV(f) = \frac{1}{b-a} \frac{1}{[f]} \int_a^b |f'(x)| dx,$$

where

$$[f] = \max_{a \leq x \leq b} f(x) - \min_{a \leq x \leq b} f(x).$$

The denominators in (5) for the domain size and range size rescale the graph of the function to “fit” into a unit square.

Without loss of generality, we make the interval symmetric around the origin: $a = -b$. Then if we get $g(x) = \alpha f(\beta x)$ with two scalars $\alpha, \beta > 0$ defined on the interval $[a/\beta, b/\beta]$, one can verify that $TV(g) = \alpha TV(f)$, but $T(g) = \beta T(f)$ due to the change of the interval size, which suggests an increasing roughness if $\beta$ is bigger than one and this index $T$ is insensitive to $\alpha$. When $\beta > 1$ and is an integer, by periodically extending the definition of $f$, we now regard $g(x) = \alpha f(\beta x)$ defined on the same $[a, b]$ as the original $f$ — a conventional setting in homogenization theory [29]. Then $TV(g) = \alpha \beta TV(f)$ and we still have $T(g) = \beta T(f)$ again since $\alpha$ is absorbed by the rescaling factor $[f]$ in the definition of (5). One more property of $T$ is the following. Assume $f$ is an even function attaining the minimum zero value at the origin in the interval

$I = [a, b] = [-l, l]$, then if $f$ is convex (or concave), we have $TV(f) = 2[f]$, and $T(f) = 1/l$. One example like this is the quadratic function $f(x) = \beta x^2/2$. If $f$ is not even, then $T$ in (5) is sensitive to the values at two endpoints.

1 The barrier is the difference in $f$ between a local minimizer and its neighboring saddle point along the transition path.

### D. Roughness index for high dimensional functions

To generalize the above 1D index $T$ to any dimension, we follow the idea of projection to randomly sampled direction with filter-wise normalization

$$f_d(s) := J(\theta + s d)$$

where $\theta$ is a given reference point and $d$ is a Gaussian random direction with zero mean and identity covariance matrix followed by filter-wise normalization [15]. The domain of $s$ is defined on a prescribed interval $[-l, l]$. By varying $l$, we can change the size of the region in concern around the reference point $\theta$. Unlike in [15] which used just one sampled direction $d$ in the visualization procedure, we consider the standard deviation of $f_d$ with respect to the randomness in the directions, so the roughness index (RI) is defined as follows

$$\mathcal{I}(\theta; \mathcal{J}) := \frac{\text{std}_d TV(f_d)}{E_d TV(f_d)}.$$  (6)

Here the standard deviation is adopted to describe the change of “roughness” across different directions. The rescaling by the expectation here is to further reduce the influence of the magnitude of $T$ values.

**Example II.1.** We examine the index by looking at a quadratic landscape $\mathcal{J}(\theta) = \frac{1}{2} \theta^T H \theta$ where the reference point is taken as the minimizer (the origin) and set the interval size $l = 1$. $H$ is a positive definite matrix. Then $f_d(s) = s d^T H d$ and $TV(f_d) = |d^T H d|$. If $d$ follows the standard Gaussian distribution with zero mean and identity covariance matrix, then by Hutchinson’s trick, $E_d TV(f_d) = E_d d^T H d = E_d \text{Tr}(dd^T H) = \text{Tr}(E_d (dd^T)) = \text{Tr}(H)$. But $T(f_d) \equiv 1$ in view of (5) and the roughness index $\mathcal{I}$ in (6) is zero for any quadratic function.

### E. Algorithm

The details of the computational procedure is as follows. Assume $\theta^*$ is an arbitrary point of interest. In many cases, we consider a minimum point obtained by minimizing the loss function $\mathcal{J}$. To calculate RI w.r.t. this point, detailed description on the numerical implementation of RI is available in Algorithm 1. The complexity is linearly proportional to $M \times m$ and independent of the dimension of $\theta$.

The number of directions $M$ and the number of partitions for interval $m$ are chosen sufficiently large in practice to make sure the numerical results are convergent. In addition, the various values of interval length $l$ are also tested for specific applications (See Remark III.1).

### III. Numerical Results

Consider the Poisson equation on $\Omega = (0, 1)^d$:

$$\begin{cases}
-\Delta u = f(x), & \text{in } \Omega, \\
u(x) = 0, & \text{on } \partial \Omega.
\end{cases}$$  (7)

The forcing term $f$ is specified by assuming the form of the solution first. For example, we assume the exact solution

$$u(x) = \prod_{i=1}^d \sin(\pi x_i), \quad x = (x_1, \cdots, x_d),$$  (8)
Algorithm 1: Computation of Roughness Index

Input: Loss $\mathcal{J}$, point $\theta^*$, number of directions $M$, interval length $l_i$, and number of step size $m$

Output: Roughness Index $I$ at $\theta^*$

1. $i \leftarrow 1$
2. while $i \leq M$ do
   3. Sample an iid standard Gaussian random direction $d_i$;
   4. Apply the filter-wise normalization for $d_i$: $\tilde{d}_i \leftarrow d_i$
   5. $j \leftarrow 0$
   6. while $j \leq m$ do
      7. Partition $[-l_i, l_i]$ into $m+1$ subintervals uniformly:
         $$s_{i,j} = -l_i + j \frac{2l_i}{m}, j = 0, 1, \ldots, m$$
      8. $j \leftarrow j + 1$
   9. end
10. Calculate the maximum and minimum along $\tilde{d}_i$:
    11. $$\mathcal{J}^i_{\max} = \max_{0 \leq j \leq m} \{ \mathcal{J}(\theta^* + s_{i,j}\tilde{d}_i) \}$$
    12. $$\mathcal{J}^i_{\min} = \min_{0 \leq j \leq m} \{ \mathcal{J}(\theta^* + s_{i,j}\tilde{d}_i) \}$$
13. Approximate normalized TV $T_i$ :
    14. $$T_i = \frac{1}{2l_i} \sum_{j=0}^{m-1} \frac{\mathcal{J}(\theta^* + s_{i,j}\tilde{d}_i) - \mathcal{J}(\theta^* + s_{i,j+1}\tilde{d}_i)}{\mathcal{J}^i_{\max} - \mathcal{J}^i_{\min}}$$
15. $i \leftarrow i + 1$
16. The roughness index $I := \sigma/\mu$, where $\mu, \sigma$ are the mean value and the standard deviation of $\{T_i\}_{i=1}^M$.

then we have $f(x) = d\pi^2 \prod_{i=1}^d \sin(\pi x_i)$. Denote

$$u(x; \theta) = \prod_{i=1}^d (x_i - 1) x_i \cdot \text{NN}(x; \theta).$$

where NN$(x; \theta)$ is a function represented by a NN. The corresponding loss functions are

$$\mathcal{J}_G(\theta) = \int_{\Omega} (-\Delta u(x; \theta) - f(x))^2 \, dx$$

for the DGM, and

$$\mathcal{J}_R(\theta) = \int_{\Omega} \left( \frac{1}{2} |\nabla u(x; \theta)|^2 - f(x) u(x; \theta) \right) \, dx$$

for the DRM, respectively.

In what follows, we use the relative $L^2$ error to measure the numerical error of solving the PDE,

$$\text{error} = \frac{\|u(x; \theta^*) - u(x)\|}{\|u(x)\|},$$

where $\| \cdot \|$ denotes the $L^2$ norm for functions of $x$, $u(x; \theta^*)$ is the DNN approximation, and $u(x)$ is the exact solution.

A. 1D Poisson equation

Consider the following 1D Poisson equation

$$\begin{cases}
-u''(x) = f(x), & x \in (0, 1), \\
u(0) = u(1) = 0.
\end{cases}$$

The exact solution is set as $u(x) = \sin \pi x$, so that $f(x) = \pi^2 \sin \pi x$. At this true solution, we have the global minima for $J_G(u(x)) = 0$, and $J_R(u(x)) = -\pi^2/4 \approx -2.4674$.

The numerical solution is in the form of $u(x; \theta) = (x-1)x \cdot \text{NN}(x; \theta)$. Various width $w$ is tested for ResNet and FCNet. The loss functions $\mathcal{J}(\theta)$ are non-convex now, but in practice one can generally find the global minima due to the perfect fitting capability of the neural network [8], [30].

The 1D integrals in (10) and (11) are approximated by a quadrature rule with $N$ uniform points on the interval $[0, 1]$. And we refer this $N$ as to the batch size since in the training we use all these $N$ points in each gradient-based iteration.

| Loss | $\theta_G$ | $\theta_R$ | $\tilde{\theta}_G$ |
|------|------------|------------|-----------------|
| $J_G(\theta)$ | 5.9933e-05 | 0.1044 | 5.7418e-05 |
| $J_R(\theta)$ | -2.4715 | -2.4716 | -2.4715 |

| Distance | $(\theta_G, \theta_R)$ | $(\theta_G, \tilde{\theta}_G)$ | $(\theta_R, \tilde{\theta}_G)$ |
|----------|----------------|----------------|----------------|
| $\|\|_2$ | 3.7243 | 3.8342 | 0.3349 |
| $\|\|_\infty$ | 2.3929 | 2.3052 | 0.2138 |

1) Local optimizers: Staring from the same initial guesses used to train $J_G$ and $J_R$, we use the full-batch gradient descent to find one local minimizer for each loss function, denoted by $\theta_G$ and $\theta_R$, respectively. Even though both parameters $\theta_G$ and $\theta_R$ gives approximate solutions to the PDE, these two parameters $\theta_G$ and $\theta_R$ are quite different. See Table II. After obtaining $\theta_G$ and $\theta_R$ from the DGM and DRM respectively, we swap them as the new initial guesses to train $J_G$ and $J_R$. This is to look for a new optimal parameter $\theta_G$ by minimizing $J_G$ with the new initial guess $\theta_R$ and for $\theta_R$ of $J_G$ in a like manner by using the initial $\theta_G$. We find $\theta_R$ is almost identical to $\theta_G$ and conclude $\theta_G$ and $\theta_R$ are minimizers of $J_G$; $\theta_G$ and $\theta_R$ ($= \theta_R$) as well as $\theta_G$ are minimizers of $J_R$.

The loss values at these points are shown in Table I.
2) Difference between DGM and DRM: We observed that the DGM generally obtains a better accuracy in solving PDE result than the DRM in our case here. We compare their accuracy by checking the PDE errors in (12) of their corresponding PDE solutions $u(\cdot; \theta_G)$ and $u(\cdot; \theta_R)$. We tested the ResNet of one block with different widths in Table III. Since the NN and the training algorithm as well as the initial guess are exactly the same, we attribute this discrepancy to the difference of loss in the DGM and DRM.

### TABLE III

| $w$  | 2    | 3    | 4    | 5    | 6    |
|------|------|------|------|------|------|
| $u(\cdot; \theta_G)$ | 5.21e-2 | 1.81e-2 | 7.12e-4 | 8.01e-8 | 8.31e-8 |
| $u(\cdot; \theta_R)$ | 1.64e-3 | 9.48e-4 | 7.63e-4 | 7.76e-4 | 6.75e-6 |

(a) The decay of loss functions.

(b) The decay of relative $L_2$ error of $u(x; \theta)$ to the true PDE solution.

Fig. 2. The loss functions and the relative $L_2$ error for ResNet with width $w = 2, 3, 4, 5, 6$. Left column: DGM; Right column: DRM.

We furthermore provide complementary results about the convergence for DGM and DRM toward $\theta_G$ and $\theta_R$ respectively. Figure 2 shows the decay of the loss and the relative $L^2$ error (12) in the training process. One interesting observation comes from the comparison of the loss and the error. The DRM is very effective to decrease the loss for all widths, but inefficient to decrease the PDE error. It seems that after the early stage of quick decay for the loss function, the DRM trajectories wander around in a neighborhood of the minimizer of the loss function in order to further reduce the PDE’s error, but with much more strenuous efforts than the DGM. As a comparison, the DGM has a better match for the decay between the PDE error and the loss function. This is easy to understand since by (10), the loss of the DGM is $J_G(u) = \int_0^1 (u'' - u''_{ex})^2 dx = \|u'' - u''_{ex}\|^2$, with the only difference of a (linear) Laplace operator, which is more closely linked to the PDE error (12) than the DRM.

3) Roughness index (RI): Now we report our main numerical results of $I$ for this 1D problem. We record roughness indices in several setting of parameter combinations. The calculation involves the minimizers of interests, the number of directions $M$, the interval length $l$, the number of points $m$ partitioned in the interval.

We first present the results of roughness indices of the DGM and the DRM around their first set of optimal parameters $\theta_G$ and $\theta_R$. With a fixed width $w = 4$, Table IV to Table VII show the comparing results of the roughness indices for the two models with various combinations of network architecture (ResNet or FCNet), the width $w$, the values of $M$, $l$ and $m$. In all cases, particularly with the ResNet architecture, we have strong numerical evidences to claim that the roughness index in the DGM is significantly smaller than that in the DRM.

### TABLE IV

| $M$ | $I_{DGM}$ | $I_{DRM}$ |
|-----|----------|----------|
| ResNet | FCNet | ResNet | FCNet |
| 50  | 0.0455  | 0.2387  | 0.4665 | 0.2472 |
| 100 | 0.0615  | 0.2157  | 0.4443 | 0.2256 |
| 150 | 0.0668  | 0.2186  | 0.4653 | 0.2195 |

### TABLE V

| $l$ | $I_{DGM}$ | $I_{DRM}$ |
|-----|----------|----------|
| $M = 100$ | $m = 100$ | $l$ |
| ResNet | FCNet | ResNet | FCNet |
| 0.00025 | 0.0287  | 0.1846  | 0.6743  | 0.2139 |
| 0.0005  | 0.0073  | 0.1336  | 0.7264  | 0.1712 |
| 0.001   | 0.0109  | 0.0731  | 0.7311  | 0.1291 |
| 0.005   | 0.0074  | 0.0253  | 0.1863  | 0.0537 |
| 0.01    | 0.0127  | 0.0157  | 0.1525  | 0.0227 |
| 0.05    | 0.0418  | 0.0553  | 0.0876  | 0.0705 |

### TABLE VI

| $l$ | $m$ | $I_{DGM}$ | $I_{DRM}$ |
|-----|-----|----------|----------|
| $M = 100$ | $l$ |
| ResNet | FCNet | ResNet | FCNet |
| 0.00005 | 0.0517 | 0.3639 |
| 0.00010 | 0.0587 | 0.4593 |
| 0.00015 | 0.0394 | 0.5709 |
| 0.00020 | 0.0353 | 0.6222 |
| 0.00025 | 0.0287 | 0.6743 |
| 0.00030 | 0.0275 | 0.7096 |

**Remark III.1.** We remark that although the choice of $M$ and $m$ is simple (the larger the better), the choice of the interval length $l$ is important and one should test a few values for this parameter. $l$ characterizes the size of a small neighborhood we
are interested when measuring the roughness. If \( l \) is too large, the domain of interest is too large to smear the roughness around the reference point. Table VIII shows such phenomena as \( l \) increases to a very large value: the disparity in the roughness index between the two models is less and less significant. The visualization plot in Figure 3 corresponds to \( l = 0.01 \). Conceptually, the suitable size of \( l \) should be comparable to the size of the basin of attraction, but here we deal with a highly non-convex landscape and it is not possible to pinpoint this value. So instead, we varied the choices of \( l \) in practice and seek for a robust result in a reasonable range of \( l \). We find \( l = 0.01 \) is quite representative for our example here.

Lastly, we report the RI for the second set of parameters \( \theta_G \). Recall that we validated \( \theta_G \) and \( \theta_G \) are two different minimizers of \( R_G \); \( \theta_R \) and \( \theta_G = \theta_R \) are two different minimizers of \( R_R \). We have reported the roughness index for \( \theta_G \) and \( \theta_R \) before. Table IX adds the RI of the DGM and DRM at all these three points. It shows that the RI of the DGM is almost equal for the DGM’s two local minimizers and this is also true for the DRM’s local minimizers. And the roughness index of the DGM is indeed much smaller than the roughness index of the DRM, regardless of which minimizer of their own is investigated. We cannot confirm that this holds for all local minimizers since it is not possible to explore all these minimizers. But we are inclined to the conjecture of a larger roughness index for the landscape of the DRM than the DGM, when the ResNet is used.

4) Validation of RI by visualization: After we calculated the numerical values of RI for the DGM and DRM models, we have reached a conclusion that the landscape of the DRM seems rougher than the DGM. To validate this claim, we apply the visualization technique in [15] to show heuristic and visual evidence.

We use visualization with filter-wise normalization in a randomly chosen 2D space. The contour plots of loss landscapes for the DGM and the DRM with ResNet and FCNet at their local minimizers \( \theta_G \) and \( \theta_R \) are shown in first two rows of Figure 3. From the comparisons between the left (DGM) and right (DRM) columns, we can heuristically see that the DGM has a relatively flat and smooth neighborhood while the DRM seems more rough and more oscillatory near \( \theta_R \). This difference remains true both for the fully-connected network and the ResNet. We change the set of optimal parameters to the second set \( \theta_G \) and \( \theta_R \) in the subfigure (c) and we still see the similar observation. Therefore, the visualization results we obtained here from random directions qualitatively confirms our conjecture that the DRM has more rough landscapes near its local minimizers, while the landscapes of the DGM at local minimizers are relatively less rough.

5) Understanding difference of the roughness index for two models: Recall the definition of roughness index, \( I = \sigma/\mu \), is the ratio of the standard deviation and the expectation of the (1D) normalized TV (5) when the loss function is projected on \( M \) random directions. After establishing that \( I \) are indeed different for the DGM and the DRM at the local minimizers \( \theta_G \) and \( \theta_R \), respectively, we want to further check whether the reason is from the standard deviation \( \sigma \) or the expectation \( \mu \). Figure 4 discovers that the difference comes from the standard deviation \( \sigma \), not the mean \( \mu \). In fact, the means of the normalized TV across different directions are almost identical in the two models. This figure strongly indicates the importance of taking account of random effect of the directions. A larger \( \sigma \) means a higher anisotropy of the loss function in the high dimension. Therefore, we can say the higher roughness of the DRM comes from the more anisotropic loss function.

Remark III.2. Note that the “anisotropy” here has nothing to do with the eigenvalues of the Hessian matrix. Some conventional literatures use the ratio of eigenvalues to represent the anisotropy for a quadratic function. However, we have known that the roughness index is null for quadratic functions. The “anisotropy” refers to the uncertainty of the TV norms (the “1D” roughness) across different directions in a high dimensional space.

6) Roughness index on gradient descent path: So far we have focused on the roughness index around the local mini-

---

**TABLE VII**

| \( w \) | \( I_{DGM} \) | \( I_{DRM} \) |
|---|---|---|
| 2  | 0.0356 | 0.0843 |
| 3  | 0.0289 | 0.2389 |
| 4  | 0.0216 | 0.0890 |
| 5  | 0.0266 | 0.0992 |
| 6  | 0.0208 | 0.0481 |

**TABLE VIII**

| \( l \) | \( I_{DGM} \) | \( I_{DRM} \) |
|---|---|---|
| 0.1 | 0.0759 | 0.1050 |
| 0.2 | 0.1151 | 0.1381 |
| 0.3 | 0.1509 | 0.1727 |
| 0.4 | 0.1562 | 0.1823 |

**TABLE IX**

| \( \theta_G \) | \( I_{DGM} \) | \( I_{DRM} \) |
|---|---|---|
| \( \theta_R \) | 0.0127 | 0.1525 |
| \( \theta_G \) | 0.1448 | 0.1732 |
| \( \theta_G \) | 0.0153 | 0.1660 |
The answer is no: the disparity of the roughness only appears near the local minimizers. We provide the evidences in the following. Firstly, we compute the RI for arbitrarily points in the parameter space by following the standard strategies such as Xavier initializations [31], and two other random samples. Table X shows that the difference in the index is very marginal. In fact, we observed from this table that the expectation $\mu$ is nearly $1/2l$ for almost every direction. This means the 1D projected loss function is monotonic in all directions at all initial points; the loss landscape is essentially non-oscillatory almost everywhere for random locations.

**TABLE X**

| Initialization | DGM Mean $\mu$ | DGM Std $\sigma$ | DRM Mean $\mu$ | DRM Std $\sigma$ |
|----------------|----------------|------------------|----------------|------------------|
| Xavier         | 50.00          | 7.141e-15        | 50.00          | 7.105e-15        |
| Uniform(-1, 1) | 50.13          | 1.340            | 50.04          | 0.3669           |
| Normal(0, 1)   | 50.00          | 1.596e-15        | 50.10          | 0.9939           |

The second evidence is from the examination of the RI along a path from an initial point to the local minimizer. We first generate and save a (gradient-descent) path obtained from the training process, then compute the roughness index at a few representative points which are ordered by the epoch. Figure 5 presents these two curves of the indices for the two models and suggests that there is a cross-over of the roughness around at the epoch 2000. Recall in Figure 2 which records the training process, the training processes in general have already approached a vicinity of the minimizer around epoch 2000 and after that the training is to mainly improve the accuracy further within this vicinity. By dividing the training process into these two stages, Figure 5 essentially tells us that in these two stages, the regions that the trajectories are exploring can be very different in terms of the roughness index.

**Fig. 4.** The mean $\mu$ and the std $\sigma$ in the roughness index $I = \sigma/\mu$ at $\theta_G$ and $\theta_R$, respectively, for various interval lengths $l$. $M = 100$, $m = 100$ and ResNet.
B. 3D equation with a low-regularity solution

To further check our conclusion, we consider a problem with a low-regularity solution over $\Omega = \{ x \in \mathbb{R}^3 : |x| < 1 \}$

\[
\begin{align*}
-\Delta u &= f(x), \quad \text{in } \Omega, \\
u(x) &= 0, \quad \text{on } \partial \Omega.
\end{align*}
\]

The exact solution $u(x) = \sin \left( \frac{\pi}{2} (1 - |x|) \right)$ is continuous but not differential at the origin. Then $f(x) = \frac{\pi^2}{4} \sin \left( \frac{\pi}{2} (1 - |x|) \right) + \frac{\pi}{|x|} \cos \left( \frac{\pi}{2} (1 - |x|) \right)$. The solution is parametrized $u(x; \theta) = (|x| - 1) \cdot NN(x; \theta)$. The ResNet is used with three residual blocks and neural width $w = 8$, thus the total number of parameters is 617. The number of epochs is 5000 and the batch size $N$ is 1000. Roughness indices at the same point $\theta_C$ are recorded in Table XI. These results point to the same conclusion we had before.

### Table XI

| $l$ | $m$ | $M$ | $I_{\text{DGM}}$ | $I_{\text{DRM}}$ |
|-----|-----|-----|-----------------|-----------------|
| 0.25 | 20 | 100 | 0.1162 | 0.1500 |
| 0.05 | 20 | 100 | 0.0770 | 0.2126 |
| 0.1 | 10 | 100 | 0.1189 | 0.1688 |
| 0.2 | 20 | 100 | 0.1292 | 0.1420 |
| 0.3 | 30 | 100 | 0.1151 | 0.1384 |

D. 1D wave equation

The last example is the wave equation in one dimension:

\[
\begin{align*}
-\Delta u &= f(x), \quad t \in [0, T], x \in (0, 1), \\
u(t, x) &= 0, \quad t \in [0, T], x = 0, 1 \\
u(0, x) &= u_t(0, x) = 0, \quad x \in (0, 1)
\end{align*}
\]

with the exact solution $u(x, t) = t^2 \sin(\pi x)$. Similarly, the solution is parametrized by the DNN approximation $u(x; \theta) = t^2 \sin(\pi x) \cdot NN(x; \theta)$. The Deep Ritz method is not applicable here because the wave equation has no variational formulation. So instead of comparing the landscapes of the DGM and the DRM, we explore the change of RI along a path from a gradient descent in training the loss function. Figure 6 presents this curve of the RI along with the value of the DGM loss. We find that the RI value along the path is quite similar to that for the DGM in Fig. 5 for the Poisson equation: the gradient descent trajectory first go through a high RI region and then gradually decreases together with the loss. Since the box size $l = 0.01$ is used here, we can say the gradients near the minimizer $\theta^*$ are all close to zero in the neighborhood with size $l$.

C. High dimensional Poisson equation

Our next example is the equation (7) when $d = 10$. The ResNet is used with three residual blocks and neural width $w = 20$, thus the total number of parameters is 3601. The number of epochs is 50000 and the batch size $N$ is 100000. The relative errors in both DGM and DRM are around 1e-3 with 50000 epochs. Roughness indices of attractor in terms of the number of random directions $M$, interval of interest, and the number of grid points, are recorded in Table XII. Again, we observe that the roughness index in the DGM is slightly smaller than that in the DRM.

### Table XII

| $l$ | $m$ | $M$ | $I_{\text{DGM}}$ | $I_{\text{DRM}}$ |
|-----|-----|-----|-----------------|-----------------|
| 0.025 | 20 | 100 | 0.1162 | 0.1500 |
| 0.05 | 20 | 100 | 0.0770 | 0.2126 |
| 0.1 | 15 | 100 | 0.1045 | 0.1420 |
| 0.2 | 20 | 100 | 0.1151 | 0.1384 |
| 0.3 | 30 | 100 | 0.1124 | 0.1284 |

Fig. 6. The roughness indices along the path generated from the training process with $l = 0.01$, $M = 100$, $m = 40$ for 1D wave equation. The ResNet is used with one residual blocks and neural width $w = 8$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
IV. CONCLUDING REMARKS

In this work, we introduce the roughness index (RI) to characterize the roughness of loss function near any reference point. Through numerous experiments, we show that this quantity is particularly useful for the high dimensional parameter space and can effectively comparing the “roughness” between two neural network landscapes arising from DGM and DRM. Our RI is based on the 1D normalized total variation in any specified region, rather than the Hessian matrix at the local minimizer as a local quadratic approximation, so this index can be applied to both convex and non-convex landscapes. Furthermore, we propose an efficient algorithm to compute RI by randomly sampling the projection directions.

In the comparison between DGM and DRM, we see significant smaller values of the roughness index in the DGM than the DRM at various local minimizers when ResNet is used. We also discover that this difference of the roughness mainly comes from the standard deviation of the directional randomness. By examining the roughness index along the optimization trajectory, we have the empirical observations that although both are initialized in a smooth region with low RI, the RI in the DRM gradually increases while the DGM has the ability to pass through a high RI region and then settle to a low RI basin of the minimizer. We conjecture that this empirical observation of RI differences in the landscape may be the reason for the performance differences of using these two models in practice to solve high dimensional PDEs, measured by the final accuracy of the numerical solution and the difficulties of training the models. The last comment is although we propose the roughness index and demonstrate its power in the background of solving PDE problems, we think this roughness concept and our method of RI are also important in studying highly non-convex landscapes for general machine-learning tasks. Particularly, the signature pattern of increasing-then-decreasing RI on the optimization path in the DGM, as shown in Figure 5 and Figure 6, implies that by following the gradient descent, the trajectory experiences the “flat-rough-flat” transition when traveling the landscape. We conjecture that this could be also valid in many machine-learning tasks such as image classification problems, but the careful empirical validations with heuristic or rigorous analysis are still yet under our investigation.

REFERENCES

[1] C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck, “An overview on deep learning-based approximation methods for partial differential equations,” arXiv preprint arXiv:2012.12348, 2020.
[2] W. E and B. Yu, “The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems,” Communications in Mathematics and Statistics, vol. 6, no. 1, pp. 1–12, 2018.
[3] T. Siringanu and K. Spiliopoulos, “DGM: A deep learning algorithm for solving partial differential equations,” Journal of Computational Physics, vol. 375, pp. 1339–1364, 2018.
[4] 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, vol. 378, pp. 686–707, 2019.
[5] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in Proceedings of COMPSTAT 2010, Y. Lechevallier and G. Saporta, Eds., 2010, pp. 177–186.
[6] A. Choromanska, Y. LeCun, and G. B. Arous, “Open problem: The landscape of the loss surfaces of multilayer networks,” in Conference on Learning Theory. PMLR, 2015, pp. 1756–1760.
[7] Q. Nguyen, M. C. Mukkamala, and M. Hein, “On the loss landscape of a class of deep neural networks with no bad local valleys,” in ICLR, 2019.
[8] A. C. Gaht and A. Walker, “The energy landscape of a simple neural network,” in 10th NIPS Workshop on Optimization for Machine Learning, 2012.
[9] G. Swirszcz, W. M. Czarnecki, and R. Pascanu, “Local minima in training of deep networks,” in ICLR, 2016.
[10] D. Dinl, 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, 2017.
[11] C. Baldassi, F. Pottorino, and R. Zecchina, “Shaping the learning landscape in neural networks around wide flat minima,” Proceedings of the National Academy of Sciences, vol. 117, no. 1, pp. 161–170, 2020.
[12] F. Draxler, K. Veschgini, M. Salmoher, and F. Hamprecht, “Essentially no barriers in neural network energy landscape,” in International conference on machine learning, PMLR, 2018, pp. 1309–1318.
[13] S. Mei, M. Ranganath, and M. I. Jordan, “A mean field view of the landscape of two-layer neural networks,” Proceedings of the National Academy of Sciences, vol. 115, no. 33, pp. E7665–E7671, 2018.
[14] A. Jacot, F. Gabriel, and C. Hongler, “Neural tangent kernel: Convergence and generalization in neural networks,” in Advances in neural information processing systems, 2018, pp. 8571–8580.
[15] H. Li, Z. Xu, G. Taylor, C. Schuler, and T. Goldstein, “Visualizing the loss landscape of neural nets,” in Advances in Neural Information Processing Systems, 2018, pp. 6389–6399.
[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” CoRR, 2015.
[17] L. Wu, Z. Zhu, and W. E, “Towards understanding generalization of deep learning: Perspective of loss landscapes,” ICMC, 2017.
[18] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.
[19] M. Hardt, B. Recht, and Y. Singer, “Train faster, generalize better: Stability of stochastic gradient descent,” in International conference on machine learning, 2016, pp. 1225–1234.
[20] N. Sharir Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima,” ICLR, Sep. 2017.
[21] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” Physica: D: Nonlinear Phenomena, vol. 60, no. 1, pp. 259–268, 1992.
[22] I. Selesnick, “Total variation denoising (an MM algorithm),” NYU Polytechnic School of Engineering Lecture Notes, vol. 32, 2012.
[23] H. Daneshmand, J. Kohler, A. Lucchi, and T. Hofmann, “Escaping saddles with stochastic gradients,” in ICML, vol. 80, 2018, pp. 1155–1164.
[24] B. Kleinberg, Y. Li, and Y. Yuan, “An alternative view: When does SGD escape local minima?” in Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 2698–2707.
[25] M. I. Freidlin and A. D. Wentzell, Random Perturbations of Dynamical Systems, 3rd ed., ser. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 2012.
[26] Q. Li, C. Tai, and W. E, “Stochastic modified equations and adaptive stochastic gradient algorithms,” in 34th International Conference on Machine Learning, 2017, pp. 3306–3314.
[27] W. Hu, Z. Zhu, H. Xiong, and J. Huan, “Quasi-potential as an implicit regularizer for the loss function in the stochastic gradient descent,” arXiv preprint arXiv:1901.06054, 2019.
[28] G. Bachmann, L. Narici, and E. Beckenstein, Fourier and wavelet analysis. Springer Science & Business Media, 2012.
[29] G. Papanicolaou, A. Bensoussan, and J. Lions, Asymptotic Analysis for Periodic Structures, ser. ISSN. Elsevier Science, 1978.
[30] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning (still) requires rethinking generalization,” Communications of the ACM, vol. 64, no. 3, pp. 107–115, 2021.
[31] 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.