Addressing Spectral Bias of Deep Neural Networks by Multi-Grade Deep Learning

Ronglong Fang, Yuesheng Xu*
Department of Mathematics and Statistics, Old Dominion University
{rfang002, y1xu}@odu.edu

Abstract

Deep neural networks (DNNs) have showcased their remarkable precision in approximating smooth functions. However, they suffer from the *spectral bias*, wherein DNNs typically exhibit a tendency to prioritize the learning of lower-frequency components of a function, struggling to effectively capture its high-frequency features. This paper is to address this issue. Notice that a function having only low frequency components may be well-represented by a shallow neural network (SNN), a network having only a few layers. By observing that composition of low frequency functions can effectively approximate a high-frequency function, we propose to learn a function containing high-frequency components by composing several SNNs, each of which learns certain low-frequency information from the given data. We implement the proposed idea by exploiting the multi-grade deep learning (MGDL) model, a recently introduced model that trains a DNN incrementally, grade by grade, a current grade learning from the residue of the previous grade only an SNN (with trainable parameters) composed with the SNNs (with fixed parameters) trained in the preceding grades as features. We apply MGDL to synthetic, manifold, colored images, and MNIST datasets, all characterized by presence of high-frequency features. Our study reveals that MGDL excels at representing functions containing high-frequency information. Specifically, the neural networks learned in each grade adeptly capture some low-frequency information, allowing their compositions with SNNs learned in the previous grades effectively representing the high-frequency features. Our experimental results underscore the efficacy of MGDL in addressing the spectral bias inherent in DNNs. By leveraging MGDL, we offer insights into overcoming spectral bias limitation of DNNs, thereby enhancing the performance and applicability of deep learning models in tasks requiring the representation of high-frequency information. This study confirms that the proposed method offers a promising solution to address the spectral bias of DNNs. The code is available on GitHub: `Addressing Spectral Bias via MGDL`.

1 Introduction

Deep neural networks (DNNs) have achieved tremendous success in various applications, including computer vision [15], natural language processing [38], speech recognition [26], and finance [27]. From a mathematical perspective, the success is mainly due to their high expressiveness, as evidenced by theoretical demonstrations showing their capability to approximate smooth functions with arbitrary precision [3, 10, 14, 17, 48]. Various mathematical aspects of DNNs as an approximation tool were recently investigated in [11, 16, 22, 25, 35, 45–47, 49]. However, it was noted in [30, 42] that the standard deep learning model, which is called single grade deep learning (SGDL) in this paper,

*Corresponding author: y1xu@odu.edu

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3625

trains a DNN end-to-end leading to learning bias towards low-frequency functions. While this bias may explain the phenomenon where DNNs with a large number of parameters can achieve low generalization error [7, 40, 41], DNNs trained by SGDL struggle to capture high-frequency components within a function even though they can well-represent its low-frequency components. This bias may potentially limit the applicability of DNNs to problems involving high-frequency features, such as image reconstruction [12, 24, 37], seismic wavefield modeling [39], high-frequency wave equations in homogenization periodic media [9], and high energy physics [28]. Especially, in medical image reconstruction such as PET/SPECT, high-frequency components play a crucial role in determining image resolution, which is critical in clinical practice, as higher resolution leads to earlier and more accurate disease diagnosis.

There have been some efforts to address this issue. A phrase shift DNN was proposed in [5], where the original dataset was first decomposed into subsets with specific frequency components, the high-frequency component was shifted downward to a low-frequency spectrum for learning, and finally, the learned function was converted back to the original high frequency. An adaptive activation function was proposed in [18] to replace the traditional activation by scaling it with a trainable parameter. A multiscale DNN was introduced in [6, 23], in which the input variable was first scaled with different scales and then the multiscale variables were combined to learn a DNN. It was proposed in [36] first to map the input variable to Fourier features with different frequencies and then to train the mapped data by DNNs. All these approaches can mitigate the spectral bias issue of DNNs to some extent.

Despite of encouraging progresses made in mitigating the spectral bias of DNNs, practical learning with DNNs remains a persistent challenge due to the bias, especially for learning from higher-dimensional data. This issue deserves further investigation. We propose to address this issue by understanding how a high-frequency function can be more accurately represented by neural networks. On one hand, it has been observed [7, 30, 31] that a function having only low frequency components can be well represented by a shallow neural network (SNN), a network having only a few layers. On the other hand, the classical Jacobi–Anger identity expresses a complex exponential of a trigonometric function as a linear combination of its harmonics that can contain significant high-frequency components. Even though the complex exponential function and the trigonometric function both are of low frequency, their composition could contain high frequency components. This motivates us to decompose a function containing high-frequencies as a *sum-composition* form of low-frequency functions. That is, we decompose it into a sum of different frequency components, each of which is further broken down to a *composition of low-frequency functions*. In implementing this idea, we find that the multi-grade deep learning (MGDL) model recently introduced in [43, 44] matches seamlessly for constructing the sum-composition form for a function of high-frequency. It is the purpose of this study to introduce the general methodology in addressing the spectral bias issue of DNNs and implement it by employing MGDL as a technical tool. We demonstrate the efficacy of the proposed approach in four experiments with one-dimensional synthetic data, two-dimensional manifold data, two-dimensional colored images, and very high-dimensional modified National Institute of Standards and Technology (MNIST) data. Our numerical results endorse that the proposed approach can effectively address the spectral bias issue, leading to substantial improvement in approximation accuracy in comparison with the traditional SGDL training approach.

Contributions of this paper include: (a) We propose a novel approach to address the spectral bias issue by decomposing a function containing high-frequencies as a sum of different frequency components, which are represented as compositions of low-frequency functions. (b) We investigate the efficacy of MGDL in decomposing a function of high-frequency into its "sum-composition" form of SNNs. (c) We successfully apply the proposed approach to synthetic data in 1 and 2 dimensions and real data in 2 and 784 dimensions, showing that it can effectively address the spectral bias issue.

2 Proposed Approach and Multi-Grade Learning Model

We introduce a novel approach to tackle the spectral bias issue and review the MGDL model.

We begin with a quick review of the definition of DNNs. A DNN is a successive composition of an activation function composed with a linear transformation. Let $\mathbb{R}$ denote the set of all real numbers, and d, s be two positive integers. A DNN with depth D consists of an input layer, $D-1$ hidden layers, and an output layer. Let $\mathbb{N}_D := \{1, 2, \ldots, D\}$. For $j \in \{0\} \cup \mathbb{N}_D$, let d_j denote the number of neurons in the j-th hidden layer with $d_0 := d$ and $d_D := s$. We use $\mathbf{W}_j \in \mathbb{R}^{d_j \times d_{j-1}}$ and $\mathbf{b}_j \in \mathbb{R}^{d_j}$

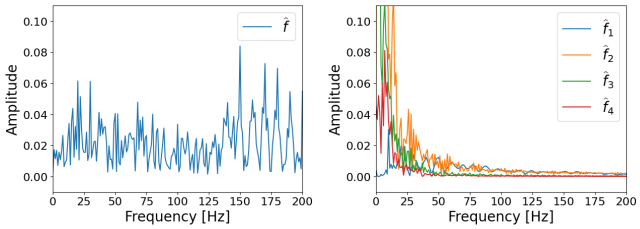


Figure 1: Spectrum comparison of $f := f_1 + f_2 \circ f_1 + f_3 \circ f_2 \circ f_1 + f_4 \circ f_3 \circ f_2 \circ f_1$ and f_j: Amplitude versus one-side frequency plots for function f (Left) and f_j for $j \in \mathbb{N}_4$ (Right). The function f is of high-frequency and functions f_j all are of low-frequency.

to represent the weight matrix and bias vector, respectively, for the j-th layer. By $\sigma : \mathbb{R} \to \mathbb{R}$ we denote an activation function. When σ is applied to a vector, it means that σ is applied to the vector componentwise. For an input vector $\mathbf{x} := [x_1, x_2, \ldots, x_d]^\top \in \mathbb{R}^d$, the output of the first layer is defined by $\mathcal{H}_1(\mathbf{x}) := \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$. For a DNN with depth $D \geq 3$, the output of the $(j+1)$-th hidden layer can be identified as a recursive function of the output of the j-th hidden layer, defined as $\mathcal{H}_{j+1}(\mathbf{x}) := \sigma(\mathbf{W}_{j+1}\mathcal{H}_j(\mathbf{x}) + \mathbf{b}_{j+1})$, for $j \in \mathbb{N}_{D-2}$. Finally, the output of the DNN with depth D is an s-dimensional vector-valued function defined by

$$\mathcal{N}_D\left(\{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^D; \mathbf{x}\right) = \mathcal{N}_D(\mathbf{x}) := \mathbf{W}_D\mathcal{H}_{D-1}(\mathbf{x}) + \mathbf{b}_D. \tag{1}$$

Suppose that data samples $\mathbb{D} := \{\mathbf{x}_\ell, \mathbf{y}_\ell\}_{\ell=1}^N$ are chosen. The loss on $\mathbb{D}$ is defined as

$$\mathcal{L}\left(\{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^D; \mathbb{D}\right) := \frac{1}{2N}\sum\nolimits_{\ell=1}^N \left\|\mathbf{y}_\ell - \mathcal{N}_D\left(\{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^D; \cdot\right)(\mathbf{x}_\ell)\right\|_2^2. \tag{2}$$

The traditional SGDL model is to minimize the loss function L defined by (2) with respect to $\Theta := \{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^D$, which yields the optimal parameters $\Theta^* := \{\mathbf{W}_j^*, \mathbf{b}_j^*\}_{j=1}^D$ and the corresponding DNN $\mathcal{N}_D(\Theta^*; \cdot)$. When D is relatively small, for example, $D < 5$, we call $\mathcal{N}_D$ an SNN. It is well-recognized that training an SNN is notably easier than training a DNN.

We motivate the proposed idea by a simple example. We consider the function $f(\mathbf{x})$, $\mathbf{x} \in [0, 1]$, whose Fourier transform is shown in Figure 1 (Left), where the Fourier transform is defined by $\hat{f}(\mathbf{t}) := \int_{-\infty}^{\infty} f(\mathbf{x})e^{-i2\pi\mathbf{t}\mathbf{x}}d\mathbf{x}$. To compute the Fourier transform of f defined on $[0, 1]$, we extend f to the entire real line by assigning its value to be zero for $\mathbf{x} \notin [0, 1]$. Observing from Figure 1 (Left), the function f has significant high-frequency components, with frequencies varying from 0 to 200. The function f can be represented as

$$f(\mathbf{x}) = f_1(\mathbf{x}) + (f_2 \circ f_1)(\mathbf{x}) + (f_3 \circ f_2 \circ f_1)(\mathbf{x}) + (f_4 \circ f_3 \circ f_2 \circ f_1)(\mathbf{x}), \quad \mathbf{x} \in [0, 1], \tag{3}$$

where $\circ$ denotes the composition of two functions. Note that the Fourier transforms $\hat{f}_j$, $j = 1, 2, 3, 4$, are displayed in Figure 1 (Right). Clearly, the functions f_j, $j = 1, 2, 3, 4$, are of low-frequency, with frequencies mainly concentrating on the interval $[0, 50]$. This example surely demonstrates that a function of high-frequency can be expressed as a sum of compositions of lower-frequency functions. This observation leads to the proposed approach of addressing the spectral bias of DNNs to be studied in this paper. The legitimacy of the proposed idea may be reinforced by the Jacobi–Anger identity [2], which expresses a complex exponential of a trigonometric function as a linear combination of its harmonics. Even though both the complex exponential function and the trigonometric function are of low-frequency, their composition contains many high-frequency components. We now review the Jacobi–Anger identity, the identity named after the 19th-century mathematicians Carl Jacobi and Carl Theodor Anger. It has the form

$$e^{ia\sin(b\mathbf{x})} = \sum\nolimits_{n=-\infty}^{\infty} J_n(a)e^{inb\mathbf{x}}, \tag{4}$$

where i denotes the imaginary unit and $J_n(a)$ denotes the n-th Bessel function of the first kind, see details in [2]. Taking the real part of the both sides of the Jacobi–Anger identity (4), we obtain that

$$\cos(a\sin(b\mathbf{x})) = \sum\nolimits_{n=-\infty}^{\infty} J_n(a)\cos(nb\mathbf{x}). \tag{5}$$

The left-hand side of (5) is a composition of two low-frequency functions $\cos(a\mathbf{x})$ and $\sin(b\mathbf{x})$, having frequencies $a/(2\pi)$ and $b/(2\pi)$, respectively, while the right-hand side is a linear combination of $\cos(nb\mathbf{x})$ with n taking all integers. The high-frequency of the composition can be estimated by a rule of thumb. Specifically, the left-hand side of (5) is a frequency-modulated sinusoidal signal

https://doi.org/10.52202/079017-3625

[32, 33], with its frequencies spreading on an interval centered at zero. It follows from the well-known Carson bandwidth rule [8, 29, 32], regarded as a rule of thumb, that more than 98% frequencies are located within the interval $[-(ab+b)/(2\pi), (ab+b)/(2\pi)]$. Therefore, the highest frequency of $\cos(a\sin(b\mathbf{x}))$ can be well-estimated by $\frac{ab+b}{2\pi}$, which is greater than the frequencies of $\cos(a\mathbf{x})$ and $\sin(b\mathbf{x})$ when $a > 0$ and $b > 1$. These suggest that a composition of two low-frequency functions may lead to a high-frequency function.

The example presented earlier, together with the Jacobi–Anger identity, inspires us to decompose a given function into a sum of different frequency components, each of which is a composition of lower-frequency functions, a decomposition similar to equation (3) for the function f represented in Figure 1 (Left). In other words, for a function g of high-frequency, we decompose it in a "sum-composition" form as

$$g = \sum\nolimits_{k=1}^{K} \bigodot\nolimits_{j=1}^{k} g_j, \tag{6}$$

where $\bigodot_{j=1}^{k} g_j := g_k \circ \cdots \circ g_2 \circ g_1$, and g_j, $j \in \mathbb{N}_k$, are all of low-frequency. The function f represented in (3) is a special example of (6). In the context of approximation by neural networks, we prefer expressing g_j by SNNs, as a function having only low-frequency components can be well-represented by an SNN. The MGDL model originated in [43, 44] furnishes exactly the decomposition (6), with each g_j being an SNN. We propose to employ MGDL to learn the decomposition (6), where the low-frequency function g_j is represented by an SNN.

It is worth explaining the motivation behind the MGDL model. MGDL was inspired by the human education system which is arranged in grades. In such a system, students learn a complex subject in grades, by decomposing it into sequential, simpler topics. Foundational knowledge learned in previous grades remains relatively stable and serves as a basis for learning in a present and future grades. This learning process can be modeled mathematically by representing a function that contains higher-frequency components by a "sum-composition" form of low-frequency functions. MGDL draws upon this concept by decomposing the learning process into multiple grades, where each grade captures different levels of complexity.

We now review the MGDL model that learns given data $\mathbb{D} := \{\mathbf{x}_\ell, \mathbf{y}_\ell\}_{\ell=1}^{N}$. Following [43], we split a DNN with depth D into L grades, with $L < D$, each of which learns an SNN $\mathcal{N}_{D_l}$, defined as (1), with depth D_l, from the residue $\{\mathbf{e}_\ell^l\}_{\ell=1}^{N}$ of the previous grade, where $1 < D_l < D$ and $\sum_{l=1}^{L} D_l = D + L - 1$. Let $\Theta_l := \left\{\mathbf{W}_j^l, \mathbf{b}_j^l\right\}_{j=1}^{D_l}$ denote the parameters to be learned in grade l. We define recursively $g_1(\Theta_1; \mathbf{x}) := \mathcal{N}_{D_1}(\Theta_1; \mathbf{x})$, $g_{l+1}(\Theta_{l+1}; \mathbf{x}) := \mathcal{N}_{D_{l+1}}(\Theta_{l+1}; \cdot) \circ \mathcal{H}_{D_l-1}(\Theta_l^*; \cdot) \circ \ldots \circ \mathcal{H}_{D_1-1}(\Theta_1^*; \cdot)(\mathbf{x})$, for $l \in \mathbb{N}_{L-1}$, and the loss function of grade l by

$$\mathcal{L}_l(\Theta_l; \mathbb{D}) := \frac{1}{2N} \sum\nolimits_{\ell=1}^{N} \left\|\mathbf{e}_\ell^l - g_l(\Theta_l; \mathbf{x}_\ell)\right\|_2^2, \tag{7}$$

where $\Theta_l^* := \left\{\mathbf{W}_j^{l*}, \mathbf{b}_j^{l*}\right\}_{j=1}^{D_l}$ are the optimal parameters learned by minimizing the loss function $\mathcal{L}_l$ with respect to Θ_l. The residues are defined by $\mathbf{e}_\ell^1 := \mathbf{y}_\ell$ and $\mathbf{e}_\ell^{l+1} := \mathbf{e}_\ell^l - g_l(\Theta_l^*; \mathbf{x}_\ell)$, for $l \in \mathbb{N}_{L-1}$, $\ell \in \mathbb{N}_N$. When minimizing the loss function $\mathcal{L}_l(\Theta_l; \mathbb{D})$ of grade l, parameters Θ_j^*, $j \in \mathbb{N}_{l-1}$, learned from the previous $l-1$ grades are all *fixed* and $\mathcal{H}_{D_{l-1}-1}(\Theta_{l-1}^*; \cdot) \circ \ldots \circ \mathcal{H}_{D_1-1}(\Theta_1^*; \cdot)$ serves as a feature or "basis". After L grades are learned, the function $\bar{g}_L$ learned from MGDL is the summation of the function learned in each grade, that is,

$$\bar{g}_L\left(\{\Theta_l^*\}_{l=1}^{L}; \mathbf{x}\right) := \sum\nolimits_{l=1}^{L} g_l(\Theta_l^*; \mathbf{x}), \tag{8}$$

where $g_l(\Theta_l^*; \mathbf{x}) := \mathcal{N}_{D_l}(\Theta_l^*; \cdot) \circ \mathcal{H}_{D_{l-1}-1}(\Theta_{l-1}^*; \cdot) \circ \ldots \circ \mathcal{H}_{D_1-1}(\Theta_1^*; \cdot)(\mathbf{x})$, and $\mathcal{H}_{D_k-1}$ for $1 \le k \le L$ and $\mathcal{N}_{D_L}$ are SNNs learned in different grades. Thus, MGDL enables us to construct the desired "sum-composition" form (6). When $L = 1$, MGDL reduces to the traditional SGDL model.

In MGDL, we use the mean squared error (MSE) loss function. It was established in [43] that when the loss function is defined by MSE, MGDL either learns the zero function or results in a strictly decreasing residual error sequence (see, Theorem 1 in Appendix A). Since the regression problems conducted in this paper naturally align with MSE losses, it is a suitable choice. In practice, MGDL can also be applied with other loss functions, such as cross-entropy loss, when solving classification problems. In MGDL, the computation cost remains relatively consistent across all grades. For $\mathbf{x}_\ell^l := \mathcal{H}_{D_{l-1}-1}(\Theta_{l-1}^*; \cdot) \circ \mathcal{H}_{D_{l-2}-1}(\Theta_{l-2}^*; \cdot) \circ \ldots \circ \mathcal{H}_{D_1-1}(\Theta_1^*; \cdot)(\mathbf{x}_\ell)$, we recursively

let $\mathbf{x}_\ell^1 := \mathbf{x}_\ell,\ \mathbf{x}_\ell^k := \mathcal{H}_{D_{k-1}-1}(\Theta_{k-1}^*;\cdot)\circ\mathbf{x}_\ell^{k-1},\ k=2,3,\ldots,n$. When training grade l, we use the output of grade $l-1$, denoted as $\mathbf{x}_\ell^l$ along the residual $\mathbf{e}_\ell^l$, which are already obtained. The training dataset in grade l consists of $\left\{(\mathbf{x}_\ell^l, \mathbf{e}_\ell^l)\right\}_{\ell=1}^N$. This dataset is used to train a new shallow network, which is independent of the previous $l-1$ grades. Moreover, $\mathbf{x}_\ell^l$ can be computed recursively, ensuring that the computation cost for each grade remains relatively consistent.

MGDL avoids training a DNN from end to end. Instead, it trains several SNNs sequentially, with the current grade making use the SNNs learned from the previous grades as a feature and composing it with a new SNN to learn the residue of the previous grade. This allows MGDL to decompose a function that contains higher-frequency in a form of (6), with g_j being a SNN learned from grade j. In this way, higher-frequency components in the data can be effectively learned in a grade-by-grade manner. Note that the training time of MGDL increases linearly with the number of grades. This makes MGDL an effective and scalable solution for tackling complex tasks. MGDL is an adaptive approach by nature. When the outcome of the present grade is not satisfactory, we can always add a new grade without changing the previous grades.

It is worth noting that while ResNet [15] also has a sum-composition form, MGDL differs from it significantly. The "Composition" for ResNet refers to composition of layers, while that for MGDL emphasizes the composition of the SNNs sequentially learned in the previous grades. Moreover, ResNet learns all parameters of the entire sum of DNNs at once, training it from end to end, whereas MGDL learns the sum incrementally, grade by grade, in each grade training an SNN composed with the feature (the composition of the SNNs learned in the previous grades). MGDL also differs from the relay backpropagation approach proposed in [34], where a DNN is divided into multiple segments, each with its own loss function. The gradients from these losses are then propagated to lower layers of their respective segments and all segments are optimized all together by minimizing the sum of the losses. While MGDL trains SNNs in a multi-grade manner, each of which learns from the residue of the previous grade, freezing the previously learned SNNs (serving as features or "bases").

MGDL is a principle applicable to various models, including standard DNNs, convolutional neural networks, and ResNet. In this paper, we demonstrate its feasibility by applying it to standard DNNs.

3 Numerical Experiments

In this section, we study MGDL empirically in addressing the spectral bias issue of SGDL. We consider four examples: Subsections 3.1, 3.2, and 3.4 investigate regression on synthetic, manifold, and MNIST data, respectively, for which the spectral bias phenomena of SGDL are identified in [30]. Section 3.3 deals with regression on colored images, which were studied in [36] by using the Fourier features to mitigate the spectral bias. Our goal is to compare the learning performance of MGDL with that of SGDL on these datasets and understand to what extent MGDL can overcome the spectral bias exhibited in SGDL.

The loss functions defined in (2) for SGDL and (7) for MGDL are used to compute the training and validation loss when $\mathbb{D}$ is chosen to be the training and validation data, respectively. We use the relative squared error (RSE) to measure the accuracy of predictions obtained from both SGDL and MGDL. Assume that $\mathcal{N}$ is a trained neural network. For a prediction value $\hat{\mathbf{y}}_\ell := \mathcal{N}(\mathbf{x}_\ell)$ at $\mathbf{x}_\ell$, we define RSE $:= \sum_{\ell=1}^N \|\hat{\mathbf{y}}_\ell - \mathbf{y}_\ell\|_2^2 / \sum_{\ell=1}^N \|\mathbf{y}_\ell\|_2^2$. When $\mathbb{D}$ represents the training, validation, and testing data, RSE is specialized as TrRSE, VaRSE, and TeRSE, respectively.

Details of the numerical experiments conducted in this section, including computational resources, the network structure of SGDL and MGDL for each example, the choice of activation function, the optimizer, parameters used in the optimization process, and supporting figures are provided in Appendix B.

3.1 Regression on the synthetic data. In this experiment, we compare the efficacy of SGDL and MGDL in learning functions of four different types of high-frequencies.

The experiment setup is as follows. Given frequencies $\kappa := (\kappa_1, \kappa_2, \ldots, \kappa_M)$ with corresponding amplitudes $\alpha := (\alpha_1, \alpha_2, \ldots, \alpha_M)$, and phases $\varphi := (\varphi_1, \varphi_2, \ldots, \varphi_M)$, we consider approximating the function $\lambda : [0,1] \to \mathbb{R}$ defined by

$$\lambda(\mathbf{x}) := \sum\nolimits_{j=1}^{M} \alpha_j \sin\left(2\pi\kappa_j \mathbf{x} + \varphi_j\right), \quad \mathbf{x} \in [0,1] \tag{9}$$

by neural networks learned with SGDL and MGDL. We consider four settings, in all of which we choose $M := 20$, $\kappa_j := 10j$ and $\varphi_j \sim \mathcal{U}(0, 2\pi)$ for $j \in \mathbb{N}_{20}$, where $\mathcal{U}$ denotes the uniform

 https://doi.org/10.52202/079017-3625

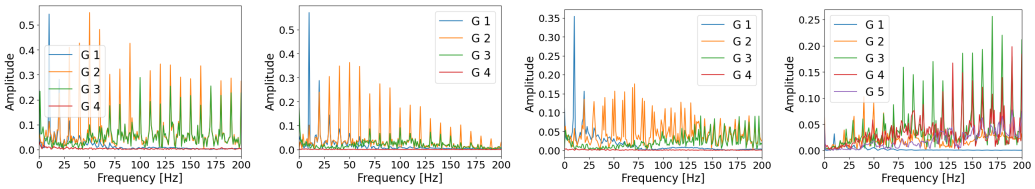


Figure 2: Amplitude versus one-side frequency plot for the learned functions learned across four grades of MGDL for settings 1-4.

Table 1: Comparison of SGDL and MGDL: Accuracy

setting	model	t_{max}	t_{min}	batch size	time(s)	TrRSE	VaRSE	TeRSE
1	SGDL	10^{-4}	10^{-4}	256	32,401	1.5×10^{-1}	1.3×10^{-1}	1.2×10^{-1}
	MGDL	10^{-4}	10^{-4}	256	27,817	$\mathbf{5.9 \times 10^{-6}}$	$\mathbf{1.9 \times 10^{-5}}$	$\mathbf{1.7 \times 10^{-5}}$
2	SGDL	5×10^{-4}	10^{-4}	256	32,053	5.8×10^{-3}	5.9×10^{-3}	5.7×10^{-3}
	MGDL	10^{-4}	10^{-4}	256	27,628	$\mathbf{1.5 \times 10^{-6}}$	$\mathbf{4.0 \times 10^{-6}}$	$\mathbf{6.5 \times 10^{-6}}$
3	SGDL	10^{-3}	10^{-4}	256	31,808	9.6×10^{-2}	8.3×10^{-2}	1.1×10^{-1}
	MGDL	10^{-4}	10^{-4}	256	24,876	$\mathbf{5.2 \times 10^{-6}}$	$\mathbf{3.2 \times 10^{-5}}$	$\mathbf{2.1 \times 10^{-5}}$
4	SGDL	5×10^{-5}	10^{-5}	256	41,063	7.9×10^{-1}	7.5×10^{-1}	7.7×10^{-1}
	MGDL	10^{-4}	10^{-4}	Full	9,875	$\mathbf{1.1 \times 10^{-3}}$	$\mathbf{2.2 \times 10^{-3}}$	$\mathbf{1.3 \times 10^{-3}}$

distribution and the random seed is set to be 0. In settings 1, 2, 3, and 4, we choose respectively the amplitudes $\alpha_j := 1$, $\alpha_j := 1.05 - 0.05j$, $\alpha_j(\mathbf{x}) := e^{-\mathbf{x}} \cos(j\mathbf{x})$, and $\alpha_j := 0.05j$, for $j \in \mathbb{N}_{20}$. Note that in setting 3, we explore the case where the amplitude varies as a function of $\mathbf{x}$ for each component. The amplitudes versus one-side frequencies for λ across the four settings are depicted in Figure 7 in Appendix B.1. For all the four settings, the training data consist of pairs $\{\mathbf{x}_\ell, \lambda(\mathbf{x}_\ell)\}_{\ell \in \mathbb{N}_{6000}}$, where $\mathbf{x}_\ell$'s are equally spaced between 0 and 1. The validation and testing data consist of pairs $\{\mathbf{x}_\ell, \lambda(\mathbf{x}_\ell)\}_{\ell \in \mathbb{N}_{2000}}$, where $\mathbf{x}_\ell$'s are generated from a random uniform distribution on $[0, 1]$, with the random seed set to be 0 and 1, respectively.

Numerical results for this example are reported in Figures 2 and 3, as well as 8 and 9 (in Appendix B.1), and Table 1. Figure 2 displays the amplitude versus the one-side frequency of the functions learned across four grades of MGDL in settings 1-3, and five grades in setting 4. In all the four settings, MGDL exhibits a pattern of learning low-frequency components in grade 1, middle-frequency components in grade 2, and high-frequency components in grades 3, 4 (and 5 for setting 4). We let $\mathcal{N}_j^* := \mathcal{N}_{D_j}(\Theta_j^*; \cdot)$ and $\mathcal{H}_j^* := \mathcal{H}_{D_j - 1}(\Theta_j^*; \cdot)$. The SNN $\mathcal{N}_1^*$ learned in grade 1 represents a low-frequency component of λ, the SNNs $\mathcal{N}_2^*$ and $\mathcal{N}_3^*$ learned in grades 2 and 3, composed with $\mathcal{H}_1^*$ and $\mathcal{H}_2^* \circ \mathcal{H}_1^*$, respectively, represents higher-frequency components of λ. Likewise, the SNN $\mathcal{N}_4^*$, learned in grade 4, composed with $\mathcal{H}_3^* \circ \mathcal{H}_2^* \circ \mathcal{H}_1^*$ represents the highest-frequency component of λ. This fact is particularly evident in setting 4 (see, the fourth subfigure in Figure 2), where the amplitude within the data increases with the frequency. For settings 1, 2, and 3, grade 4 does not learn much. This is because for the functions of these three settings, the amplitudes of higher-frequencies are *proportionally* small. However, grade 4 is still important for these settings. As shown in Figure 3 (right), grade 4 reduces the loss from 10^{-2} to 10^{-5} for setting 1 and from 10^{-4} to 10^{-6} for settings 2 and 3. This indicates that if we want a high precision, we need to include grade 4. For setting 4, we need grade 5 to learn its highest frequency component. These findings suggest that MGDL is particularly well-suited for learning the high-frequency components of the function.

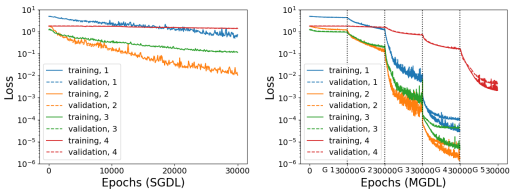


Figure 3: Comparison of SGDL (left) and MGDL (right): training and validation loss across settings 1-4.

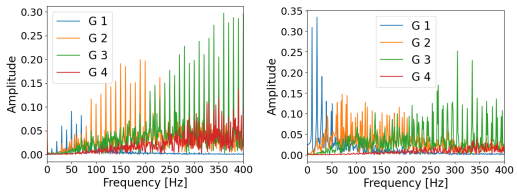


Figure 4: Amplitude vs. one-side frequency plot for the learned function across four grades of MGDL: settings 1 (left) and 2 (right) with $q = 0$.

Figure 3 compares the progress of the training and validation losses against the number of training epochs for SGDL and MGDL across the four settings. We observe that when learning a task involving high-frequency components by SGDL, the training and validation losses decrease slowly due to the spectral bias of DNN. While the same task is learned by MGDL, the learning process progresses through distinct grades. In grade 1, MGDL primarily learns low-frequency, resulting in a slow decrease in loss. In grade 2, the training loss and validation loss both decrease more rapidly due to the use of the composition of SNN $\mathcal{N}_2^*$ with the feature $\mathcal{H}_1^*$, facilitating in learning high-frequency features. This accelerated learning aspect of MGDL is further evidenced in grades 3 and 4 (as well as grade 5 for setting 4). Table 1 compares the accuracy achieved by SGDL and MGDL. Within a comparable or even less training time, MGDL increases accuracy, measured by TeRSE from 10^{-1} to 10^{-5}, 10^{-3} to 10^{-6}, 10^{-1} to 10^{-5}, and 10^{-1} to 10^{-3} in settings 1, 2, 3 and 4, respectively. Across the four settings, TeRSE values are reduced by a factor of $592 \sim 7,058$. These comparisons highlight MGDL's advantage in effectively learning high-frequency oscillatory functions.

Figure 8 in Appendix B.1 depicts the functions, in the Fourier domain, learned by SGDL (row 1) and MGDL (row 2) across the four settings, demonstrating that MGDL has a substantial reduction in the 'spectral bias' exhibited in SGDL. This is because high-frequency components are learned in a higher grade, where they are represented as the composition of a low-frequency component with the low-frequency components learned in the previous grades, and each grade focuses solely on learning a low-frequency component by an SNN. We also include Figure 9 in Appendix B.1 to compare the spectrum evolution between SGDL (1st row) and MGDL (2nd row) across settings 1-4. Notably, although in iterations SGDL and MGDL both learn low-frequency components first and then followed by middle and high-frequency components, MGDL learns in grade by grade, exhibiting significant outperformance.

3.2 Regression on the manifold data. The second experiment compares regression by SGDL and MGDL on two-dimensional manifold data, studied in [30] but with twice higher frequencies.

The goal of this experiment is to explore scenarios where data lies on a lower-dimensional manifold embedded within a higher-dimensional space. Such data is commonly referred to as manifold data [4]. Let γ be an injective mapping from $[0,1]^m \to \mathbb{R}^d$ with $m \leq d$ and $\mathcal{M} := \gamma([0,1]^m)$ denote the manifold data. A target function $\tau : \mathcal{M} \to \mathbb{R}$ defined on the manifold can be identified with function $\lambda := \tau \circ \gamma$ defined on $[0,1]^m$. Regressing the target function τ is therefore equivalent to finding $f : \mathbb{R}^d \to \mathbb{R}$ such that $f \circ \gamma$ matches λ. Following [30], we set $m := 1$, $d := 2$, and choose the mapping γ as

$$\gamma_q(\mathbf{x}) := [1 + \sin(2\pi q \mathbf{x})/2]\,(\cos(2\pi\mathbf{x}), \sin(2\pi\mathbf{x}))\,, \quad \mathbf{x} \in [0,1], \tag{10}$$

for a nonnegative integer q. Clearly, $\gamma_q : [0,1] \to \mathbb{R}^2$, and $\mathcal{M} := \gamma_q([0,1])$ defines the manifold corresponding to a flower-shaped curve with q petals when $q > 0$, and a unit circle when $q = 0$. Suppose that $\lambda : [0,1] \to \mathbb{R}$ is the function defined by (9). Our task is to learn a DNN $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f \circ \gamma_q$ matches λ. We consider two settings for λ. In settings 1 and 2, we choose $\alpha_j := 0.025j$ and $\alpha_j(\mathbf{x}) := e^{-\mathbf{x}} \cos(j\mathbf{x})$ for $j \in \mathbb{N}_{40}$, respectively. For both the settings, we choose $\kappa_j := 10j$ and $\varphi_j \sim \mathcal{U}(0, 2\pi)$ for $j \in \mathbb{N}_{40}$ with the random seed set to be 0, and consider the cases where $q := 4$ and $q := 0$. Note that the smaller q is, the more difficult the learning task is. The training data consists of pairs $\{\gamma_q(\mathbf{x}_\ell), \lambda(\mathbf{x}_\ell)\}_{\ell \in \mathbb{N}_{12000}}$, where $\mathbf{x}_\ell$'s are equally spaced between 0 and 1. The validation and testing data consist of pairs $\{\gamma_q(\mathbf{x}_\ell), \lambda(\mathbf{x}_\ell)\}_{\ell \in \mathbb{N}_{4000}}$, where $\mathbf{x}_\ell$'s are generated from a random uniform distribution on $[0,1]$, with random seed set to be 0 and 1, respectively.

Numerical results for this example are reported in Figures 4-5, and 10 (in Appendix B.2), and Table 2. Figure 4 illustrates the frequency of functions learned across four grades of MGDL for settings 1 and 2, where $q := 0$. In both of the settings, MGDL exhibits a pattern of learning low-frequency components in grade 1, middle-frequency components in grade 2, and high-frequency components in grades 3 and 4. Therefore, the high-frequency components within the function mainly learned in higher grades, in which the learned function is a composition of the SNNs learned from several grades. That is, MGDL decomposes a high-frequency component as the composition of several lower-frequency components, facilitating effectively learning high frequency features within the data.

Table 2 compares the approximation accuracy achieved by SGDL and MGDL for settings 1 and 2. For SGDL, reducing the value of q makes the learning task for both settings more challenging, due to the spectral bias of DNNs. When $q := 4, 0$ in setting 1 and $q := 0$ in setting 2, learning becomes especially challenging for SGDL. In such cases, MGDL significantly outperforms SGDL by achieving higher accuracy in approximately half to one-third of the training time for both settings. Figure 5 displays the training and validation loss for SGDL and MGDL. Figure 10 illustrates the

 https://doi.org/10.52202/079017-3625

Table 2: Accuracy comparison: SGDL versus MGDL.

setting	q	method	t_{max}	t_{min}	batch size	time (s)	TrRSE	VaRSE	TeRSE
1	4	SGDL	10^{-4}	10^{-6}	1024	$28,832$	2.7×10^{-1}	2.4×10^{-1}	2.8×10^{-1}
		MGDL	10^{-3}	10^{-4}	Full	$15,519$	$\mathbf{4.9 \times 10^{-5}}$	$\mathbf{1.8 \times 10^{-4}}$	$\mathbf{1.1 \times 10^{-4}}$
	0	SGDL	10^{-4}	10^{-5}	1024	$29,051$	5.5×10^{-1}	5.4×10^{-1}	5.4×10^{-1}
		MGDL	10^{-3}	10^{-4}	Full	$15,969$	$\mathbf{1.9 \times 10^{-4}}$	$\mathbf{2.7 \times 10^{-4}}$	$\mathbf{2.1 \times 10^{-4}}$
2	4	SGDL	10^{-3}	10^{-6}	512	$44,083$	2.0×10^{-4}	1.8×10^{-3}	2.3×10^{-4}
		MGDL	10^{-3}	10^{-4}	Full	$11,067$	$\mathbf{8.5 \times 10^{-6}}$	$\mathbf{1.4 \times 10^{-3}}$	$\mathbf{4.2 \times 10^{-5}}$
	0	SGDL	10^{-4}	10^{-4}	512	$41,941$	1.1×10^{-2}	1.8×10^{-2}	1.0×10^{-2}
		MGDL	10^{-3}	10^{-4}	Full	$11,027$	$\mathbf{3.7 \times 10^{-5}}$	$\mathbf{2.2 \times 10^{-3}}$	$\mathbf{8.7 \times 10^{-5}}$

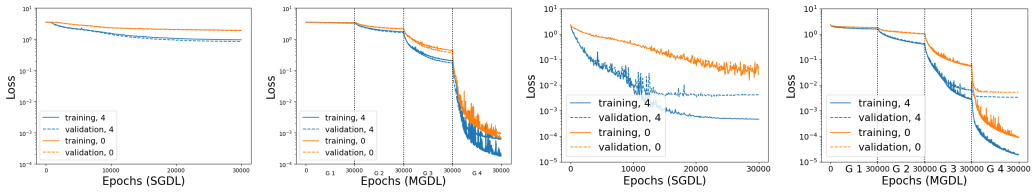


Figure 5: Comparison of the training and validation loss for SGDL (1st and 3rd subfigures) and MGDL (2nd and 4th subfigures) in settings 1 (1st and 2nd subfigures) and 2 (3rd and 4th subfigures).

spectrum evolution throughout the learning process for settings 1 and 2. We observe that MGDL outperforms SGDL in accuracy during the learning process. Table 2 and Figure 10 consistently demonstrate that MGDL exhibits a substantial advancement in addressing the spectral bias.

3.3 Regression on two-dimensional colored Images. In the third experiment, we compare performance of MGDL and SGDL for regression of two-dimensional color images.

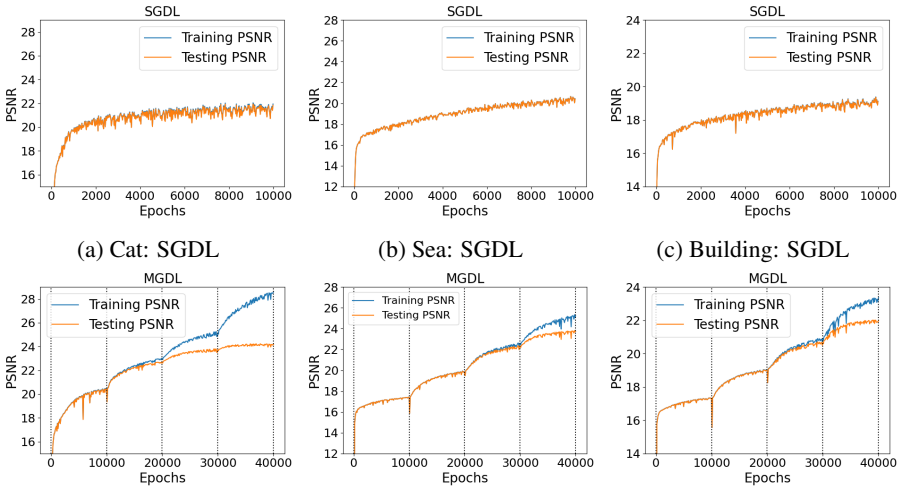
We test the models with the 'cat' image from website Cat, and the 'sea' and 'building' images from the Div2K dataset [1]. The input to the models is the pixel coordinates, and the output is the corresponding RGB values. The training dataset consists of a regularly spaced grid containing $1/4$ of the image pixels, while the test dataset contains the full image pixels. We use peak signal-to-noise ratio (PSNR) defined as in (12) to evaluate the accuracy of images obtained from MGDL and SGDL. PSNR values computed over the train and test dataset are denoted by TrPSNR and TePSNR, respectively.

Numerical results for this experiment are presented in Table 3, Figure 6, and Figures 11-13 (in Appendix B.3). Table 3 compares the PSNR values of images obtained from MGDL and SGDL. For MGDL, it is evident that adding more grades consistently improves the image quality, as both training and testing PSNR values increase substantially with the addition of each grade across all images Cat, Sea, and Building. MGDL outperforms SGDL by $2.35 \sim 3.93$ dB for the testing PSNR values. Specifically, for images Cat, Sea and Building, MGDL surpasses SGDL by 2.35, 3.93 and 2.88 dB, respectively. This demonstrates the superiority of MGDL in representing images compared to SGDL. Figure 6 illustrates the training and testing PSNR values during the training process for the three images. It is evident that MGDL facilitates a smoother learning process as more grades are added, in comparison to SGDL. This observation aligns with the results presented in Table 3.

Predictions of each grade of MGDL and of SGDL are illustrated in Figures 11, 12, and 13 for images Cat, Sea, and Building, respectively. In all cases, grade 1 captures only the rough outlines of the images, representing the lower-frequency components. As we progress to grades 2, 3, and 4, more details corresponding to the higher-frequency components are added. This demonstrates the effectiveness of MGDL in progressively learning high-frequency features within the images. Moreover, the image quality achieved with MGDL is notably superior to that with SGDL.

3.4 Regression on MNIST data. In our fourth experiment, we apply MGDL to regress on high-dimension data, the MNIST data. We compare the training and validation loss, and TeRSE for SGDL and MGDL, when learning high-frequency features from the data.

We set up this experiment following [30], with a focus on comparing performance of SGDL and MGDL in learning high-frequency features. We choose the handwriting digits from MNIST dataset [21], composed of 60,000 training samples and 10,000 testing samples of the digits "0" through



(a) Cat: SGDL (b) Sea: SGDL (c) Building: SGDL

(d) Cat: MGDL (e) Sea: MGDL (f) Building: MGDL

Figure 6: Comparison of PSNR values for SGDL and MGDL on images Cat, Sea, and Building: SGDL (a)-(c), MGDL (d)-(f).

Table 3: PSNR comparison: SGDL versus MGDL.

image	method	grade	learning rate	time (s)	TrPSNR	TePSNR
cat	MGDL	1	10^{-3}	38	20.45	20.41
		2	10^{-3}	40	22.97	22.67
		3	5×10^{-4}	40	25.14	23.71
		4	5×10^{-4}	28	28.59	24.18
	SGDL		5×10^{-3}	77	21.93	21.83
sea	MGDL	1	10^{-2}	160	18.62	18.62
		2	10^{-3}	173	21.57	21.50
		3	10^{-3}	174	24.17	23.42
		4	10^{-3}	182	27.31	24.32
	SGDL		10^{-3}	685	20.43	20.39
building	MGDL	1	5×10^{-3}	171	17.30	17.29
		2	5×10^{-3}	181	18.97	18.95
		3	10^{-3}	182	20.86	20.67
		4	10^{-3}	182	23.36	21.97
	SGDL		10^{-3}	742	19.12	19.09

"9". We represent each digit $j = 0, 1, \ldots, 9$ by a one-hot vector $e_{j+1} \in \mathbb{R}^{10}$, whose $(j+1)$-th component is one and all others are zero, and denote by $\tau_0 : \mathbb{R}^{784} \to \mathbb{R}^{10}$ the classifier, which is a piecewise constant function defined by $\tau_0(\mathbf{x}) := e_{j+1}$ if $\mathbf{x}$ represents the digit j. We split the available training samples to form the training and validation data, denoted as $\mathbb{D}_0 := \{\mathbf{x}_\ell, \tau_0(\mathbf{x}_\ell)\}_{\ell \in \mathbb{N}_{n_{train}}}$ and $\mathbb{D}'_0 := \{\mathbf{x}'_\ell, \tau_0(\mathbf{x}'_\ell)\}_{\ell \in \mathbb{N}_{n_{val}}}$ respectively, with $n_{train} := 45,000$ and $n_{val} := 15,000$, and use the testing samples as the testing data, denoted as $\mathbb{D}''_0 := \{\mathbf{x}''_\ell, \tau_0(\mathbf{x}''_\ell)\}_{\ell \in \mathbb{N}_{n_{test}}}$ with $n_{test} := 10,000$. Clearly, $\mathbb{D}_0$, $\mathbb{D}_0{}'$ and $\mathbb{D}''$ are subsets of $\{[0,1]^{784}, \{e_{j+1}\}_{j=0}^{9}\}$. Letting $\psi_\kappa(\mathbf{x}) := \sin(2\pi\kappa\|\mathbf{x}\|_2)$, corresponding to a radial wave defined on the input space $\mathbb{R}^{784}$, we define the target function by $\tau_{\beta,\kappa}(\mathbf{x}) := \tau_0(\mathbf{x})\,(1 + \beta\psi_\kappa(\mathbf{x}))$, where κ is the frequency of the wave and β is the amplitude. Note that τ_0 and $\beta\psi_\kappa$ contribute respectively the lower-frequency and high-frequency components (regarded as noise) of the target function $\tau_{\beta,\kappa}$, as discussed in [30]. The modified training and validation data denoted by $\mathbb{D}_{\beta,\kappa} := \{\mathbf{x}_\ell, \tau_{\beta,\kappa}(\mathbf{x}_\ell)\}_{\ell \in \mathbb{N}_{n_{train}}}$ and $\mathbb{D}'_{\beta,\kappa} := \{\mathbf{x}'_\ell, \tau_{\beta,\kappa}(\mathbf{x}'_\ell)\}_{\ell \in \mathbb{N}_{n_{val}}}$, respectively, are used to train DNNs. Our goal is to use SGDL and MDGL to regress the modified data $\mathbb{D}_{\beta,\kappa}$ through minimizing their respective training loss, to compare their robustness to noise. The training loss is evaluated on $\mathbb{D}_{\beta,\kappa}$ and validation loss is on $\mathbb{D}'_{\beta,\kappa}$. TrRSE, VaRSE, and TeRSE are evaluated on $\mathbb{D}_{\beta,\kappa}$ $\mathbb{D}'_{\beta,\kappa}$, and $\mathbb{D}''_0$, respectively, noting that $\mathbb{D}''_0$ are test data without noise.

https://doi.org/10.52202/079017-3625

Table 4: Accuracy comparison: SGDL versus MGDL with $\beta = 1$.

κ	method	time (s)	TrRSE	VaRSE	TeRSE
1	SGDL	$3,298$	$\mathbf{3.1 \times 10^{-1}}$	4.1×10^{-1}	1.1×10^{-1}
	MGDL	$3,109$	3.5×10^{-1}	$\mathbf{3.9 \times 10^{-1}}$	$\mathbf{8.0 \times 10^{-2}}$
5	SGDL	$3,333$	$\mathbf{3.0 \times 10^{-1}}$	4.1×10^{-1}	1.1×10^{-1}
	MGDL	$3,461$	3.5×10^{-1}	$\mathbf{3.9 \times 10^{-1}}$	$\mathbf{7.8 \times 10^{-2}}$
10	SGDL	$3,199$	$\mathbf{3.1 \times 10^{-1}}$	4.1×10^{-1}	1.0×10^{-1}
	MGDL	$3,448$	3.5×10^{-1}	$\mathbf{3.9 \times 10^{-1}}$	$\mathbf{7.8 \times 10^{-2}}$
50	SGDL	$3,168$	$\mathbf{3.0 \times 10^{-1}}$	4.1×10^{-1}	1.1×10^{-1}
	MGDL	$3,484$	3.5×10^{-1}	$\mathbf{3.9 \times 10^{-1}}$	$\mathbf{7.9 \times 10^{-2}}$

We choose the amplitude β from $\{0.5, 1, 3, 5\}$. For each β, we vary the frequency κ from $\{1, 3, 5, 7, 10, 50\}$. These choices of β and κ result in functions $\tau_{\beta,\kappa}$ that have higher frequencies than those studied in [30]. Figures 14 and 15 (in Appendix B.4) compare the training and validation loss versus training time of SGDL (structure (13)) and MGDL (structures (14) and (15) respectively) for different values of frequency κ and amplitude β (depicted in the figure). We observe that for small β, for example, $\beta = 0.5$, the results of the two models are comparable. When $\beta = 1$, the training loss for SGDL keeps decreasing as training time increases, while the validation loss initially decreases and starts to increase after training of 1,600 seconds, and unlike SGDL, both the training loss and validation loss for MGDL keep decreasing. This trend indicates that over-fitting phenomenon occurs for SGDL and suggests that MGDL has a superior generalization capability. Further increasing the value of β, for example, $\beta = 3$ and $\beta = 5$, the over-fitting phenomenon occurs in both of the models. An increase in β corresponds to a higher proportion of high-frequency components within $\tau_{\beta,\kappa}$. This increased proportion of the high-frequency component significantly impacts the validation loss of SGDL, a trend also observed in MGDL. However, the effect is comparatively less with MGDL than with SGDL. We present in Table 4 TrRSE, VaRSE, and TeRSE of SGDL (structure (13)) and MGDL (structure (14)) for $\beta := 1$, where the results are obtained by choosing $t_{min} := 10^{-5}$ and $t_{max} := 10^{-4}$, and the batch size to be 'Full', for all cases of κ and for both SGDL and MGDL. We observe that while TrRSE of SGDL is smaller than that of MGDL, both VaRSE, and TeRSE of SGDL are larger than the corresponding values of MGDL, suggesting occurrence of overfitting with SGDL. MGDL's improvement in accuracy, measured by TeRSE, is about $27 \sim 29\%$. This experiment reveals that MGDL is a promising model for learning higher-dimensional data with a higher proportion of high-frequency features.

4 Conclusion

By observing that a high-frequency function may be well-represented by composition of several lower-frequency functions, we have proposed a novel approach to learn such a function. The proposed approach decomposes the function into a sum of multiple frequency components, each of which is compositions of lower-frequency functions. By leveraging the MGDL model, we can express high-frequency components by compositions of multiple SNNs of low-frequency. We have conducted numerical studies of the proposed approach by applying it to one-dimensional synthetic data, two-dimensional manifold data, colored images, and higher-dimensional MNIST data. Our studies have concluded that MGDL can effectively learn high-frequency components within data. Moreover, the proposed approach is easy to implement and not limited by dimension of the input variable. Consequently, it offers a promising approach to learn high-frequency features within (high-dimensional) dataset.

Limitation: Mathematical understanding of the spectral bias of DNNs is absent. Theoretical foundation for MGDL to address the spectral bias issue needs to be established. Numerical studies are preliminary, limited to four examples. More extensive numerical experiments will be conducted in our future work.

Acknowledgments: Y. Xu is indebted to Professor Wei Cai of Southern Methodist University for insightful discussion of the spectral bias of DNNs. Y. Xu is supported in part by the US National Science Foundation under grant DMS-2208386, and by the US National Institutes of Health under grant R21CA263876.

References

[1] Agustsson, E., & Timofte, R. (2017) Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 126-135.

[2] Arfken, G. B., Weber, H. J., & Harris, F. E. (2011) *Mathematical Methods for Physicists: A Comprehensive Guide.* Orlando, FL: Academic press.

[3] Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., ... & Lacoste-Julien, S. (2017) A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233-242.

[4] Bengio, Y., Courville, A.C., & Vincent, P. (2012) Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(8): 1798-1828.

[5] Cai, W., Li, X., & Liu, L. (2020) A phase shift deep neural network for high frequency approximation and wave problems. *SIAM Journal on Scientific Computing*, **42**(5): A3285-A3312.

[6] Cai, W., & Xu, Z. Q. J. (2019) Multi-scale deep neural networks for solving high dimensional pdes. *arXiv preprint arXiv:1910.11710.*

[7] Cao, Y., Fang, Z., Wu, Y., Zhou, D., & Gu, Q. (2019) Towards understanding the spectral bias of deep learning. *International Joint Conference on Artificial Intelligence*, pp. 2205–2211.

[8] Carson, J. R. (1922) Notes on the theory of modulation. *Proceedings of the Institute of Radio Engineers* **10**(1): 57-64.

[9] Craster, R. V., Kaplunov, J., & Pichugin, A. V. (2010) High-frequency homogenization for periodic media. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **466**(2120): 2341-2362.

[10] Cybenko, G. (1989) Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, **2**(4): 303-314.

[11] Daubechies, I., DeVore, R., Foucart, S., Hanin, B., & Petrova, G. (2022) Nonlinear approximation and (deep) ReLU networks. *Constructive Approximation*, **55**(1): 127-172.

[12] Fang, R., Xu, Y., & Yan, M. (2024) Inexact fixed-point proximity algorithm for the ℓ_0 sparse regularization problem. *Journal of Scientific Computing*, **100**(2): 58.

[13] Glorot, X., & Bengio, Y. (2010, March) Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249-256.

[14] Goodfellow, I., Bengio, Y., & Courville, A. (2016) *Deep Learning*. Cambridge, MA: MIT Press.

[15] He, K., Zhang, X., Ren, S., & Sun, J. (2016) Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.

[16] Huang, J., Jiao, Y., Li, Z., Liu, S., Wang, Y., & Yang, Y. (2022) An error analysis of generative adversarial networks for learning distributions. *Journal of Machine Learning Research*, **23**(116): 1-43.

[17] Hornik, K., Stinchcombe, M., & White, H. (1989) Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**(5): 359-366.

[18] Jagtap, A. D., Kawaguchi, K., & Karniadakis, G. E. (2020) Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, **404**: 109136.

[19] Jiang, J., & Xu, Y. (2024) Deep neural network solutions for oscillatory Fredholm integral equations. *Journal of Integral Equations and Applications* **36**(1): 23-55.

[20] Kingma, D.P., & Ba, J. (2015) Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

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

 https://doi.org/10.52202/079017-3625

[22] Li, Y., Lu, S., Mathé, P., & Pereverzev, S. V. (2024). Two-layer networks with the ReLU^k activation function: Barron spaces and derivative approximation. *Numerische Mathematik*, **156**(1): 319-344.

[23] Liu, Z., Cai, W., & Xu, Z. Q. J. (2020) Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains. *Communications in Computational Physics*, **28**(5): 1970–2001.

[24] Lu, Y., Shen, L., & Xu, Y. (2010) Integral equation models for image restoration: high accuracy methods and fast algorithms. *Inverse Problems*, **26**(4): 045006.

[25] Mhaskar, H. N., & Poggio, T. (2016) Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, **14**(06): 829-848.

[26] Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., & Shaalan, K. (2019) Speech recognition using deep neural networks: A systematic review. *IEEE Access*, **7**: 19143-19165.

[27] Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020) Deep learning for financial applications: A survey. *Applied Soft Computing*, **93**: 106384.

[28] Perkins, D. H. (2000) *Introduction to High Energy Physics*. Cambridge: Cambridge University Press.

[29] Pieper, R. J. (2001) Laboratory and computer tests for Carson's FM bandwidth rule. In *Proceedings of the 33rd Southeastern Symposium on System Theory (IEEE)*, pp., 145-149.

[30] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., & Courville, A. (2019) On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301-5310.

[31] Ronen, B., Jacobs, D., Kasten, Y., & Kritchman, S. (2019) The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32.

[32] Schulze, J.P., Kraus, M., Lang, U., & Ertl, T. (2003) Integrating pre-integration into the shear-warp algorithm. In *IEEE VGTC / Eurographics International Symposium on Volume Graphics*, pp. 109-118.

[33] Shanmugam, K. S. (1979) *Digital and Analog Communication Systems*. New York: Wiley.

[34] Shen, L., Lin, Z., & Huang, Q. (2016) Relay backpropagation for effective learning of deep convolutional neural networks. In *European Conference on Computer Vision*, pp. 467-482.

[35] Shen, Z., Yang, H., & Zhang, S. (2022) Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157: 101-135.

[36] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., ... & Ng, R. (2020) Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, pp. 7537-7547.

[37] Tang, X., Schmidtlein, C. R., Li, S., & Xu, Y. (2021) An integral equation model for PET imaging. *International Journal of Numerical Analysis and Modeling*, **18**(6): 834-864.

[38] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017) Attention is all you need. *Advances in Neural Information Processing Systems*, pp. 5998-6008.

[39] Wu, T., & Xu, Y. (2022) Inverting incomplete Fourier transforms by a sparse regularization model and applications in seismic wavefield modeling. *Journal of Scientific Computing*, **92**(2): 48.

[40] Xu, Z. J. (2018) Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*.

[41] Xu, Z. Q. J., Zhang, Y., & Luo, T. (2022) Overview frequency principle/spectral bias in deep learning. *arXiv preprint arXiv:2201.07395*.

[42] Xu, Z. Q. J., Zhang, Y., & Xiao, Y. (2019) Training behavior of deep neural network in frequency domain. In *International Conference on Neural Information Processing*, pp. 264-274.

[43] Xu, Y. (forthcoming) Multi-grade deep learning. *Communications on Applied Mathematics and Computation*. *arXiv preprint arXiv:2302.00150* published on February 1, 2023.

[44] Xu, Y. (2023) Successive affine learning for deep neural networks. *ArXiv, abs/2305.07996*, July 11, 2023.

[45] Xu, Y. & Zhang, H. (2022) Convergence of deep convolutional neural networks. *Neural Networks*, **153**: 553–563.

[46] Xu, Y. & Zhang, H. (2024) Convergence of deep ReLU networks. *Neurocomputing* **571**: 127174.

[47] Xu, Y., & Zhang, H. (2024) Uniform convergence of deep neural networks with Lipschitz continuous activation functions and variable widths. *IEEE Transactions on Information Theory*, **70**: No. 10, October 2024.

[48] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021) Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM* **64**(3): 107-115.

[49] Zhou, D. X. (2020) Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, **48**(2): 787-794.

A Analysis of the Multi-Grade Deep Learning Model

It was established in [43] that when the loss function is defined in terms of the mean square error, a grade of MGDL either learns the zero function or results in a strictly decreasing residual error sequence.

Theorem 1. *Let* $\mathbf{D}$ *be a compact subset of* $\mathbb{R}^s$ *and* $L_2(\mathbf{D}, \mathbb{R}^t)$ *denote the space of* t*-dimensional vector-valued square integral functions on* $\mathbf{D}$*. If* $\mathbf{f} \in L_2(\mathbf{D}, \mathbb{R}^t)$*, then for all* $i = 1, 2, \ldots$*,*

$$\mathbf{f} = \sum_{l=1}^{i} \mathbf{f}_l + \mathbf{e}_i, \quad \mathbf{f}_l := \mathcal{N}_l^* \circ \mathcal{N}_{l-1}^* \circ \cdots \circ \mathcal{N}_1^*.$$

where $\mathcal{N}_l^*$ *is the SNN learned in grade* l *of MGDL, and for* $i = 1, 2, \ldots$*, either* $\mathbf{f}_{i+1} = \mathbf{0}$ *or*

$$\left\|\mathbf{e}_{i+1}^*\right\| < \left\|\mathbf{e}_i^*\right\|.$$

All numerical examples presented in this paper validate Theorem 1.

B Supporting material for Section 3

We provide in this appendix details for the numerical experiments in Section 3, including computational resources, network structures of SGDL and MGDL, the choice of activation function, as well as the optimizer and parameters used in the optimization process, and some supporting figures.

The experiments conducted in Sections 3.1, 3.2, and 3.4 of Section 3 were performed on X86_64 server equipped with an Intel(R) Xeon(R) Gold 6148 CPU @ 2.4GHz (40 slots) or Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.30GHz (32 slots). In contrast, the experiments described in Section 3.3 of Section 3 were performed on X86_64 server equipped with AMD 7543 @ 2.8GHz (64 slots) and AVX512, 2 x Nvidia Ampere A100 GPU.

We choose ReLU as the activation function as in [30] for all the four experiments. For each experiment, for SGDL we test several network structures and choose the one produces the best performance. We then design MGDL, having the same total number of layers and the same number of neurons for each layer as the chosen SGDL structure, but their parameters are trained in multiple grades as described in Section 2. Details of the network structure will be described for each experiment.

The optimization problems for both SGDL and MGDL across the four experiments are solved by the Adam method [20] with 'Xavier' initialization [13]. In Sections 3.1, 3.2, and 3.4, the learning rate t_k for the k-th epoch decays exponentially with each epoch [19], calculated as $t_k := t_{max} e^{-\gamma k}$, where $\gamma := (1/K)\ln(t_{max}/t_{min})$ represents the decay rate, with K being the total number of training epochs, t_{max} and t_{min} denoting the predefined maximum and minimum learning rates, respectively. In Section 3.3, we employed a fixed learning rate for both SGDL and MGDL, as numerical results indicate that the exponential decay learning rate performs poorly. We observe that when a network structure of SGDL is split into multiple grades of MGDL, the combined computing time required to train all grades in MGDL, each for K epochs, is comparable to the computing time needed to train the SGDL with K epochs, due to only SNNs involved in MGDL. Therefore, in all the four experiments, we train SGDL and all grades of MGDL for the same K epochs. We test SGDL and MGDL with the same set of the algorithm parameters, including t_{min}, t_{max}, batch size, and K, for all the experiments. Optimal parameters were selected based on the lowest VaRSE value for Sections 3.1 and 3.2, the highest PSNR value for Section 3.3, and the lowest validation loss for Section 3.4, within the range of parameters to be described for each example.

 https://doi.org/10.52202/079017-3625

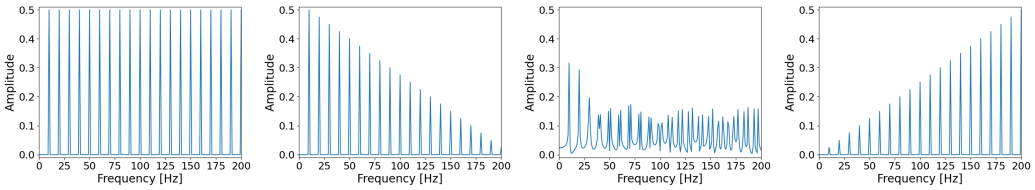


Figure 7: Amplitude versus one-side frequency plots for λ across settings 1-4.

B.1 Section 3.1

For settings 1, 2, and 3, the network structure for SGDL is

$$[1] \to [256] \times 8 \to [1], \tag{11}$$

where $[n] \times N$ indicates N hidden layers, each with n neurons. The network structure for each grade of MGDL is given by

$$\begin{aligned}
&\text{Grade 1:} \quad [1] \to [256] \times 2 \to [1] \\
&\text{Grade 2:} \quad [1] \to [256]_F \times 2 \to [256] \times 2 \to [1] \\
&\text{Grade 3:} \quad [1] \to [256]_F \times 4 \to [256] \times 2 \to [1] \\
&\text{Grade 4:} \quad [1] \to [256]_F \times 6 \to [256] \times 2 \to [1].
\end{aligned}$$

Here, $[n]_F$ indicates a layer having n neurons with parameters, trained in the previous grades, remaining fixed during the training of the current grade. We employ this notation across the numerical experiment section without further mentioning. For setting 4, the most challenging case, we employ a deeper structure for SGDL, with two more hidden layers in addition to (11). Correspondingly, for MGDL we add one more grade:

$$\text{Grade 5:} \quad [1] \to [256]_F \times 8 \to [256] \times 2 \to [1].$$

We now describe the search range of the parameters for all the four settings. We let $I_1 := \left\{10^{-4}, 10^{-5}, 10^{-6}\right\}$, $I_2 := \left\{10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}\right\}$, $I_3 := \left\{10^{-5}, 10^{-6}, 10^{-7}\right\}$ and $I_4 := \left\{5 \times 10^{-5}, 10^{-5}\right\}$. Then for both SGDL and MGDL, we test the pair (t_{min}, t_{max}) from all possible cases in the set $(I_1 \times I_2) \cup (I_3 \times I_4)$. The batch size is chosen from $256, 512$ or the full gradient (denoted by 'Full') for each epoch. The total epoch number K is set to be 30,000.

The supporting figures for this experiment include:

- Figure 7, presenting amplitude versus one-side frequency plots for λ across settings 1-4;
- Figure 8, comparing the amplitude of one-side frequency for SGDL and MGDL across settings 1-4;
- Figure 9, illustrating the evolution of spectrum comparison between SGDL and MGDL for settings 1-4.

In Figure 9, the colors in these subfigures show the measured amplitude of the network spectrum at the corresponding frequency, normalized by the amplitude of λ at the same frequency. The colorbar is clipped between 0 and 1, indicating approximation accuracy from the worst to the best when changing from 0 to 1.

B.2 Section 3.2

The network structure that we use for SGDL is

$$[2] \to [256] \times 8 \to [1],$$

and the grade network structure for MGDL is

$$\begin{aligned}
&\text{Grade 1:} \quad [2] \to [256] \times 2 \to [1] \\
&\text{Grade 2:} \quad [2] \to [256]_F \times 2 \to [256] \times 2 \to [1] \\
&\text{Grade 3:} \quad [2] \to [256]_F \times 4 \to [256] \times 2 \to [1] \\
&\text{Grade 4:} \quad [2] \to [256]_F \times 6 \to [256] \times 2 \to [1].
\end{aligned}$$

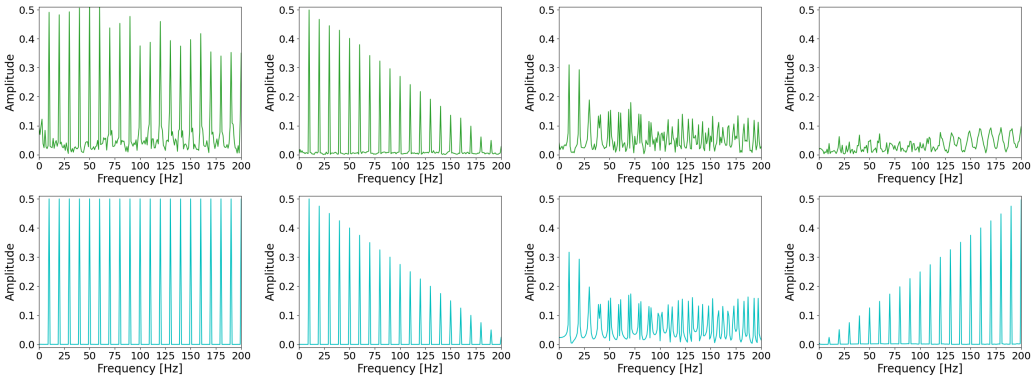


Figure 8: Comparison of SGDL (1st row) and MGDL (2nd row): Amplitude across settings 1-4.

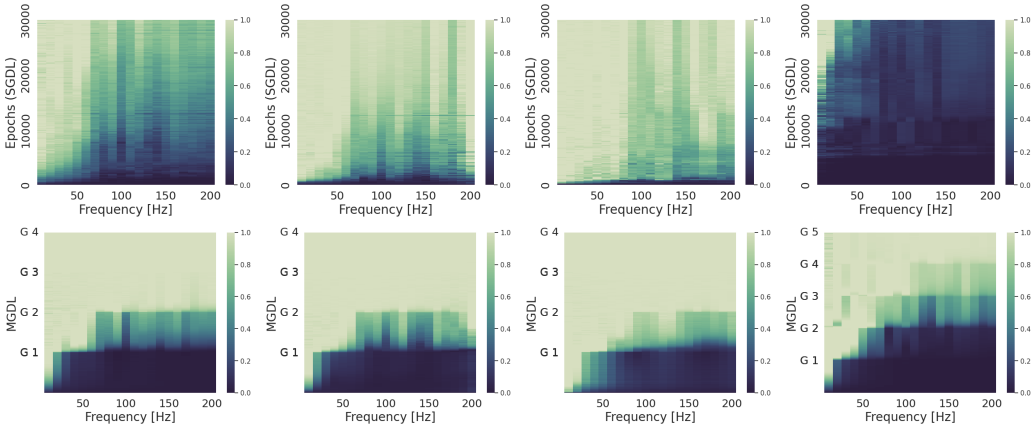


Figure 9: Comparison of SGDL (1st row) and MGDL (2nd row) for settings 1-4 of Section 3.1: The evolution of spectrum.

For choices of t_{min} and t_{max}, we let $I_1 := \{10^{-4}, 10^{-5}, 10^{-6}\}$ and $I_2 := \{1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}\}$. For both SGDL and MGDL, we test the pair (t_{min}, t_{max}) from all possible cases in the set $I_1 \times I_2$, the batch size is chosen from $512, 1024$, or the full gradient for each epoch, and the total number of the epochs K is set to be 30,000.

The supporting figures for this experiment include Figure 10, illustrating the evolution of spectrum comparison between SGDL and MGDL for settings 1-2 of Section 3.2. The mean of colorbar in Figure 10 is consistent with that in Figure 9.

B.3 Section 3.3

The network structure for SGDL is

$$[2] \to [256] \times 12 \to [3]$$

and that for MGDL is

$$\begin{aligned}
&\text{Grade 1: } [2] \to [256] \times 3 \to [3] \\
&\text{Grade 2: } [2] \to [256]_F \times 3 \to [256] \times 3 \to [3] \\
&\text{Grade 3: } [2] \to [256]_F \times 6 \to [256] \times 3 \to [3] \\
&\text{Grade 4: } [2] \to [256]_F \times 9 \to [256] \times 3 \to [3]
\end{aligned}$$

For SGDL and all grades of MGDL, we select the learning rate from the set $\{10^{-2}, 5\times10^{-3}, 10^{-3}, 5\times 10^{-4}, 10^{-4}\}$, choose the full gradient for each epoch, and set the total epoch number K to be $10,000$.

https://doi.org/10.52202/079017-3625

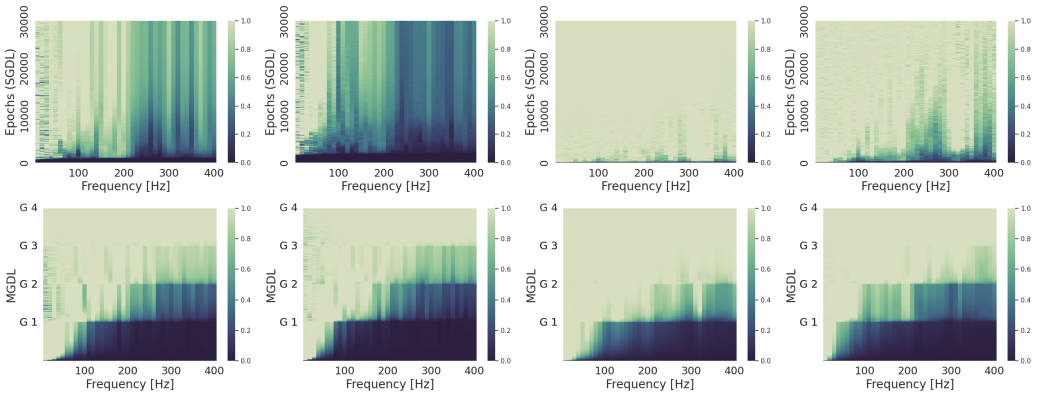
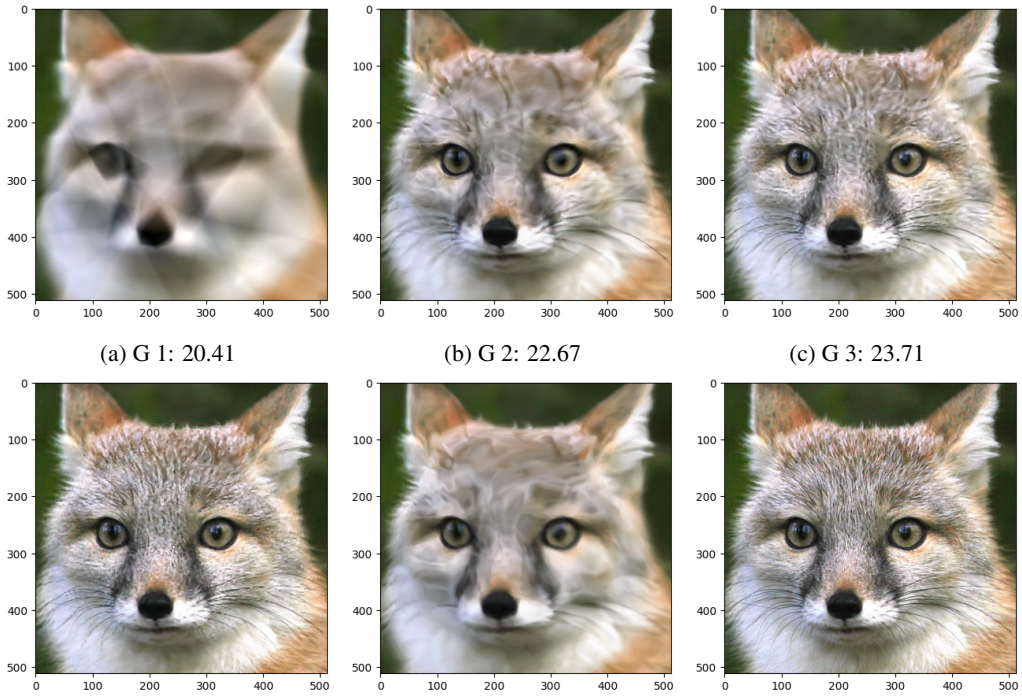


Figure 10: Comparsion of SGDL (1st row) and MGDL (2nd row) for settings 1 and 2 of Section 3.2: The evolution of spectrum (the first and second columns for the learned functions on manifolds γ_q with $q = 4$ and $q = 0$, respectively, for setting 1, while the third and fourth columns for setting 2).



(a) G 1: 20.41 (b) G 2: 22.67 (c) G 3: 23.71

(d) G 4: 24.18 (e) SGDL: 22.84 (f) Ground Truth

Figure 11: Comparison of MGDL and SGDL for image Cat. (a)-(d): Predictions of MGDL for grades 1-4, with the corresponding testing PSNR values indicated in the subtitles. (e): Prediction of SGDL with testing PSNR displayed in the subtitle. (f): Ground truth image

The quality of the reconstructed image in Section 3.3 is evaluated by the peak signal-to-noise ratio (PSNR) defined by

$$\text{PSNR} := 10 \log_{10} \left(\frac{n \times 255^2}{\|\mathbf{v} - \hat{\mathbf{v}}\|_{\text{F}}^2} \right) \tag{12}$$

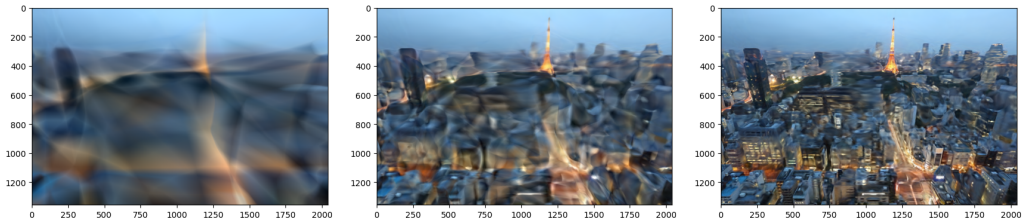
where $\mathbf{v}$ is the ground truth image, $\hat{\mathbf{v}}$ is reconstructed image, n is the number of pixels in $\mathbf{v}$, and $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm of a matrix.

The supporting figures for this experiment include Figures 11-13, which display the predicted image for MGDL and SGDL corresponding to the Cat, Sea, and Building images, respectively.

(a) G 1: 18.62 (b) G 2: 21.50 (c) G 3: 23.42

(d) G 4: 24.32 (e) SGDL: 20.39 (f) Ground Truth

Figure 12: Comparison of MGDL and SGDL for image Sea. (a)-(d): Predictions of MGDL for grades 1-4, with the corresponding testing PSNR values indicated in the subtitles. (e): Prediction of SGDL with testing PSNR displayed in the subtitle. (f): Ground truth image



(a) G 1: 17.29 (b) G 2: 18.95 (c) G 3: 20.67

(d) G 4: 21.97 (e) SGDL: 19.09 (f) Ground Truth

Figure 13: Comparison of MGDL and SGDL for image Building. (a)-(d): Predictions of MGDL for grades 1-4, with the corresponding testing PSNR values indicated in the subtitles. (e): Prediction of SGDL with testing PSNR displayed in the subtitle. (f): Ground truth image

https://doi.org/10.52202/079017-3625

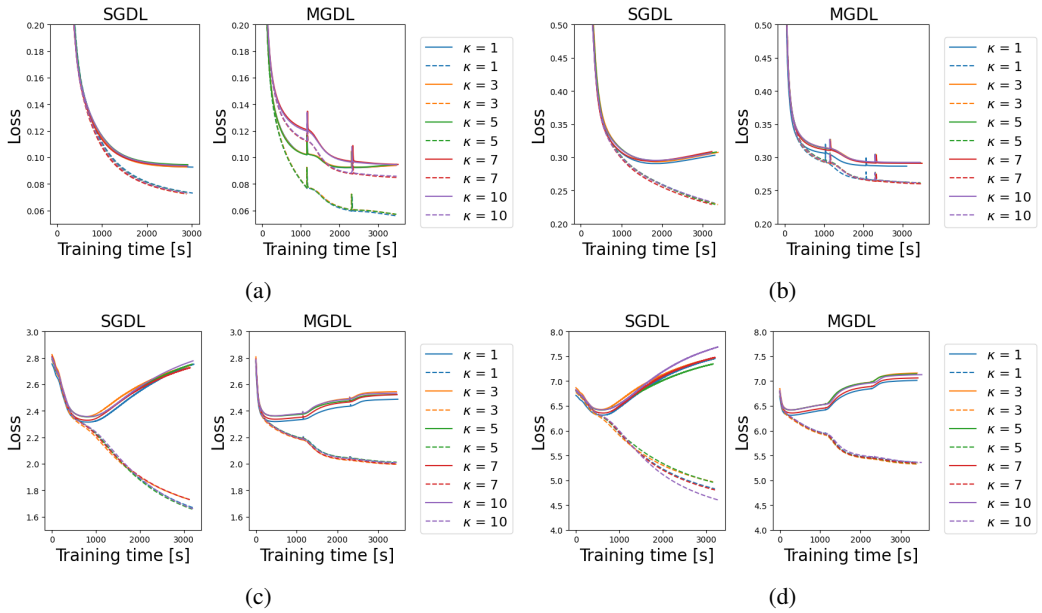


Figure 14: Comparison of SGDL and MGDL with structure (14): training (dash curve) and validation (solid curve) loss versus training time for various values of β and κ: (a) $\beta = 0.5$, (b) $\beta = 1$, (c) $\beta = 3$, (d) $\beta = 5$.

B.4 Section 3.4

The network structure for SGDL is

$$[784] \to [128] \times 6 \to [10]. \tag{13}$$

For MGDL, we consider two grade splittings. In the first splitting, we split network (13) into three grades, each with two hidden layers. The structure of MGDL for this splitting is as follows:

$$\begin{aligned} &\text{Grade 1: } [784] \to [128] \times 2 \to [10] \\ &\text{Grade 2: } [784] \to [128]_F \times 2 \to [128] \times 2 \to [10] \\ &\text{Grade 3: } [784] \to [128]_F \times 4 \to [128] \times 2 \to [10]. \end{aligned} \tag{14}$$

In the second splitting, we split network (13) into six grades, with each grade containing one hidden layer. The structure for MGDL for this case is as follows:

$$\begin{aligned} &\text{Grade 1: } [784] \to [128] \to [10] \\ &\text{Grade 2: } [784] \to [128]_F \to [128] \to [10] \\ &\text{Grade 3: } [784] \to [128]_F \times 2 \to [128] \to [10] \\ &\text{Grade 4: } [784] \to [128]_F \times 3 \to [128] \to [10] \\ &\text{Grade 5: } [784] \to [128]_F \times 4 \to [128] \to [10] \\ &\text{Grade 6: } [784] \to [128]_F \times 5 \to [128] \to [10] \end{aligned} \tag{15}$$

For choices of parameters t_{min} and t_{max}, we let $I_1 := \{10^{-4}, 10^{-5}\}$ and $I_2 := \{10^{-3}, 10^{-4}\}$. For both SGDL and MGDL, we test (t_{min}, t_{max}) from all possible cases in the set $I_1 \times I_2$, choose the batch size from 512, 1024, or the full gradient for each epoch, and set the total number of the epochs K to be 2,000.

The supporting figures for this experiment include Figure 14, which compares the training and validation loss for SGDL and MGDL with structure (14); Figure 15, which compares the training and validation loss for SGDL and MGDL with structure (15).

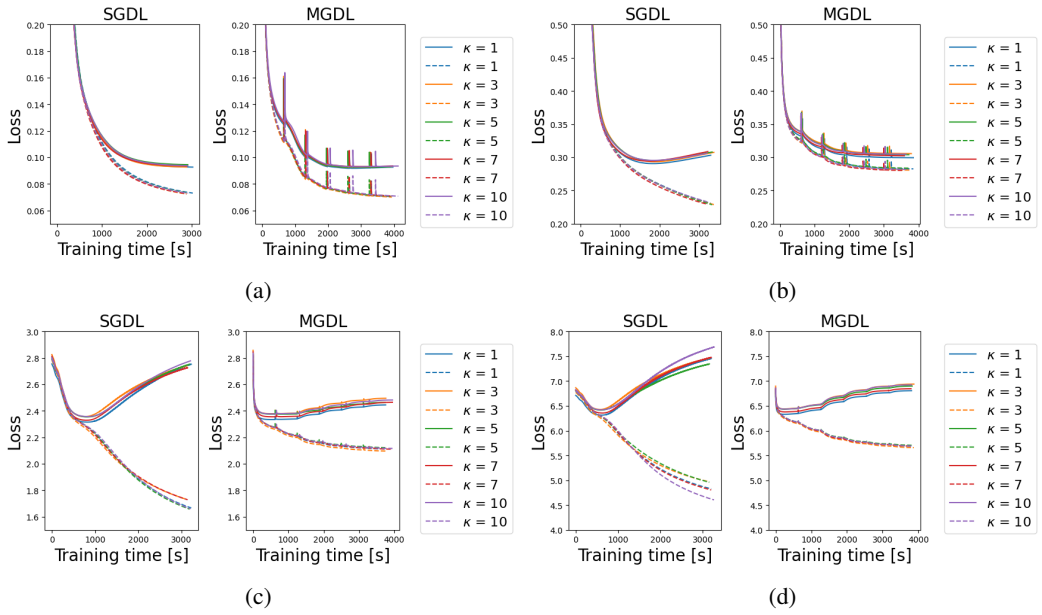


Figure 15: Comparison of SGDL and MGDL with structure (15): training (dash curve) and validation loss (solid curve) versus training time for varies values of β and κ: (a) $\beta = 0.5$, (b) $\beta = 1$, (c) $\beta = 3$, (d) $\beta = 5$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim the paper's contributions accurately and clearly in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations after the conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings,

https://doi.org/10.52202/079017-3625

model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The loss functions for SGDL and MGDL are provided. In each experiment, we clearly state experimental settings, including how to choose training/validation/testing data. The structure of SGDL and MGDL, and the parameters required in training are provided in B. The computer code is available online through the GitHub link provided in the abstract.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The computer code is available online through the GitHub link provided in the abstract. A README.txt file containing guidelines for using the code for each example is also provided.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

 https://doi.org/10.52202/079017-3625

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The loss functions for SGDL and MGDL are provided. In each experiment, we clearly state experimental settings, including how we choose training/validation/testing data. The structure of SGDL and MGDL, and the parameters required in training are provided in B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provide tables of relative mean square error accuracy and PSNR values, along with figures showing the training/validation loss and PSNR values throughout the training process.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide the experiments compute resources in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is conducted under the guidance of Code Of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the societal impacts in the conclusion section.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.

https://doi.org/10.52202/079017-3625

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We utilize the Div2K and MNIST datasets and properly cite their creators.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-3625