Enhancing the trainability and expressivity of deep MLPs with globally orthogonal initialization

Hanwen Wang
Graduate Group in Applied Mathematics
and Computational Science
University of Pennsylvania
Philadelphia, PA 19104
wangh19@sas.upenn.edu

Isabelle Crawford-Eng
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104
icraw@seas.upenn.edu

Paris Perdikaris
Department of Mechanical Engineering
and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104
pgp@seas.upenn.edu

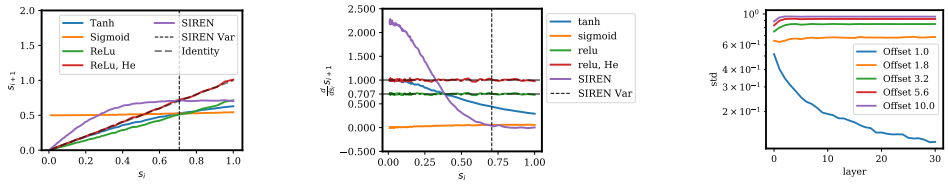
Abstract

Multilayer Perceptrons (MLPs) defines a fundamental model class that forms the backbone of many modern deep learning architectures. Despite their universality guarantees, practical training via stochastic gradient descent often struggles to attain theoretical error bounds due to issues including (but not limited to) frequency bias, vanishing gradients, and stiff gradient flows. In this work we postulate that many of such issues find origins in the initialization of the network's parameters. While the initialization schemes proposed by Glorot *et al.* and He *et al.* have become the de-facto choices among practitioners, their goal to preserve the variance of forward- and backward-propagated signals is mainly achieved by assumptions on linearity, while the presence of nonlinear activation functions may partially destroy these efforts. Here, we revisit the initialization of MLPs from a dynamical systems viewpoint to explore why and how under these classical scheme, the MLP could still fail even at the beginning. Drawing inspiration from classical numerical methods for differential equations that leverage orthogonal feature representations, we propose a novel initialization scheme that promotes orthogonality in the features of the last hidden layer, ultimately leading to more diverse and localized features. Our results demonstrate that network initialization alone can be sufficient in mitigating frequency bias and yields competitive results for high-frequency function approximation and image regression tasks, without any additional modifications to the network architecture or activation functions.

1 Introduction

It is now well understood that frequency bias [1, 2, 3] can prevent MLPs from approximating multi-scale target functions, such as the ones that routinely arise when modeling physical and engineering systems governed by differential equations [4, 5]. While many novel architectures such as residual networks [6], Fourier feature networks [7, 8], and SIREN networks [9] have been proposed to enhance the trainability and expressivity of MLPs, less attention has been placed on the importance of network initialization, especially in the context of frequency bias. Previous studies from Glorot *et al.*, He

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.



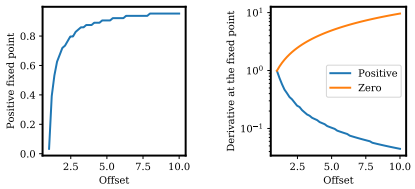
(a) (b) (c)

Figure 1: The evolution of s_i at Glorot initialization. The vertical dashed line labeled as "SIREN Var" is the standard deviation of the post-activation output proposed by *Sitzmann et al.* [9] that is invariant when $s_i = \sqrt{0.5}$ under some scaling conditions. (a) Eq. (2.1) for different activation functions f_is. (b) The derivative of Eq. (2.1) for different activation functions f_is. (c) The standard deviation of the post-activation hidden layer outputs for hyperbolic tangent networks, assuming zero mean and unit variance input.

et al. and Sitzmann *et al.* [10, 9, 11] have focused on studying how the input signal variance can be preserved from one layer to another, and deduce initialization schemes for the network weights to achieve so. More recently, Cyr *et al.* proposed the Box initialization scheme [12] for the robust initialization of MLPs with piece-wise linear activation functions, which preserves diversity in the last layer features, while Singh *et al* [13] use an additional regularization term to promote diversity in the first layer features. Inspired by these studies, we propose to model the initialization of deep MLPs as a discrete dynamical system, study its stability, and deduce a novel initialization scheme that prevents vanishing information flow and collapsed outputs.

2 Variance flow as a dynamical system

Consider a linear layer $Y = XW$, where X is the input vector, Y is the output vector, and W is the weight matrix. W assume that the entries of W are independent and identically distributed (i.i.d) with zero mean and standard deviation $c/\sqrt{d_{in}}$, where d_{in} is the input dimension, while any bias parameters are initialized to zero. In the following text, c will be denoted as a "variance offset" or the "scale" of the initialization of W. Then the output variance of the linear layer then takes the form $\text{Var}(X_i \cdot W_{j,i}) = c^2\mathbb{E}(X_i^2)/d_{in}$, while the mean is $\mathbb{E}(X_i \cdot W_{j,i}) = 0$. Moreover, the inner product $Y_k = \langle X, W_{\cdot,k}\rangle = \sum_{i=1}^{d_{in}} X_i W_{i,k}$ converges in distribution to a normal distribution with zero mean and variance $c^2\mathbb{E}(X_i^2)$ as $d_{in} \to \infty$, by the Lindeberg–Lévy Central Limit Theorem [14], if $\mathbb{E}(X_i^2)$ is finite. Consequently, we can use a normal distribution to approximate pre-activation distribution, and estimate the corresponding post-activation second moment as



(a) (b)

Figure 2: Bifurcation of Eq. (2.1) at different variance offset values. (a) The fixed point of Eq. (2.1) at different variance offset. (b) The derivatives of Eq. (2.1) at the two fixed points.

$$s_{i+1} = \sqrt{\mathbb{E}(f_i(cs_i\mathcal{N}(0,1))^2)}, \tag{2.1}$$

where f_i is the activation function, and s_i is the square root of the second moment of the post activation, of the ith layer. The system described by Eq. 2.1 can be simulated numerically for common choices of activation functions, as well as back-propagation (see Appendix B). Fig. 1 confirms the known behavior of vanishing second moment for hyperbolic tangent and ReLU activation functions, when $c = 1$ as the conventional Glorot scheme [10] initializes, eventually leading to collapsed outputs especially for deeper MLPs (see Fig. 1(c)). It also confirms the preservation of the second moment in ReLU networks initialized by the He scheme [11] where $c = \sqrt{2}$.

To further illustrate the effect of the variance offset c, we analyze the fixed points of Eq. (2.1) for hyperbolic tangent networks, since the symmetry of $\tanh$ activation allows us to replace the second moment in Eq. (2.1) with variance, and also replace s_i with standard deviation. As illustrated in Fig. 2(a), our numerical results suggest that a bifurcation occurs when varying the variance offset c. We numerically locate the new fixed point and estimate its stability, and Fig. 2(b) shows that

the larger variance offset c leads to two fixed points, and the only stable one is the positive fixed point, to which the variance of the hidden layer output with slightly larger variance offset converges (see Fig. 1(c)). Clearly, an intervention that controls the onset of this bifurcation could lead to a drastic change to the network's behavior and help prevent vanishing information flows that leads to collapsed outputs. In the next section we propose an initialization procedure that automatically tunes the layer-wise variance offset and bias, ensuring a diverse set of features in the network's last layer prior to the training.

3 Globally Orthogonal (GO) initialization of MLPs

In the previous section we show that MLP initializations that do not carefully consider the effect of non-linear activations, will likely suffer from collapsing outputs. To mitigate this adverse effect here we propose to use orthogonality in the last layer features as a measure to quantify the expressivity of an MLP representation. This is motivated by the fact that orthogonality can quantify dissimilar behavior in the feature functions. The concept of orthogonality is also widely adopted for representing functions in low-dimensional spaces among many classical numerical methods for solving differential equations by projecting the target function onto an orthogonal basis (e.g. spectral methods [15]).

Given a MLP f_w, with m hidden layers, and weights w, we use $\boldsymbol{g}_w$ to denote the post-activation outputs of the last hidden layer with d features. Letting $\boldsymbol{g}_w^j$ denote the jth feature, the cosine of the angle $\theta^{i,j}$ between $\boldsymbol{g}_w^i, \boldsymbol{g}_w^j$ is defined as

$$\cos\theta^{i,j} := \frac{\langle \boldsymbol{g}_w^i, \boldsymbol{g}_w^j \rangle}{||\boldsymbol{g}_w^i||_2 ||\boldsymbol{g}_w^j||_2}. \tag{3.1}$$

In practice, given input data $X = (x_1, ..., x_N)$, the inner product $\langle \cdot, \cdot \rangle$ is evaluated with respect to the empirical data distribution $p(x) = \frac{1}{N}\sum_{i=1}^{N} \delta(x - x_i)$, with Dirac delta function δ. Please note that we normalize the inner product of the features with the length of the features on multiple data points to avoid the scenario when the orthogonality loss on features with sharp angle that should otherwise be large remains to be small because of their lengths. Cosine of angles between the last layer features functions are then transformed to a loss objective, by which the initialization variances and the biases of the each layer except the last layer can be calibrated. Specifically, let the entries of the weight matrix of each layer be initialized with standard deviation $c_j/\sqrt{d_{in}^j}, j = 1, ..., m$, and let the bias of each layer be initialized as $b_j \mathbb{1}, j = 1, ..., m$, we proceed by optimizing $\{\boldsymbol{c}, \boldsymbol{b}\}$ by minimizing the following regularized loss

$$\min_{\boldsymbol{c},\boldsymbol{b}} -\frac{\sum_{i \neq j} \log(1 - (\cos\theta^{i,j})^2)}{d(d-1)} + \lambda ||\boldsymbol{c}||_2^2. \tag{3.2}$$

Intuitively, this orthogonality objective aims to reduce the correlation between the gradients of the parameters at different locations and therefore drastically changes the expressiveness of an MLP at initialization, as shown in Fig. 3. As opposed to previous studies that focus on the parameters orthogonality [16, 17], we directly optimize the orthogonality of the features in the function space, and therefore name it as global orthogonality objective. In the next section, we will empirical analyze the capability of GO initialization to overcome the frequency bias from the perspective of the Neural Tangent Kernel.

Empirical NTK analysis: The NTK for input locations x, y can be expressed as

$$\boldsymbol{K}(x, y) = \nabla_w f_w(x) \cdot \nabla_w f_w(y), \tag{3.3}$$

where ∇_w denotes the gradient with respect to the parameter w. Several studies [18, 19, 20] have shown that the NTK converges to a deterministic kernel as the width of the network increases to infinity. Under this setting, the MLP approximation can be asymptotically viewed as a kernel regression defined by the NTK. Furthermore, Geifman *et al.*[21] show that MLPs with ReLU activations relate closely to the Laplace kernel. We therefore believe that the behavior of the NTK at the initialization could be a representative characteristic. A visualization of the resulting NTK matrix of a 512×8 MLP reveals the correlation between the predicted MLP outputs corresponding to different input locations when responding to a small perturbation in the network's parameters. We densely sample the nearby points at different input locations in the domain to approximate the kernel functions at each input point. The kernel functions are then normalized to have unit area. Fig. 4

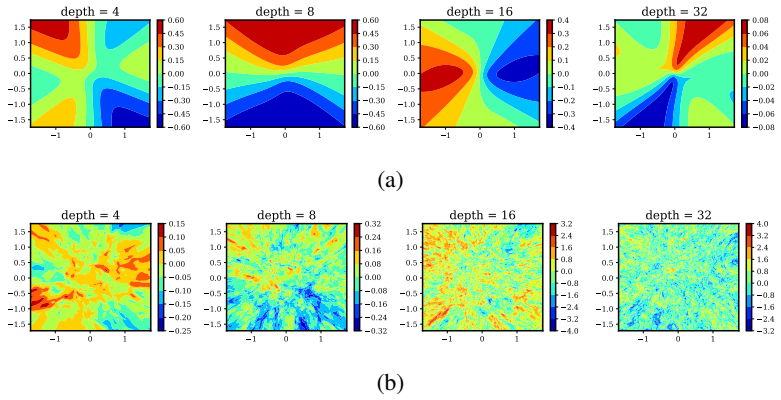


Figure 3: Neural network prediction at initialization with increasing depth (from left to right). (a) Glorot initialization. (b) Globally Orthogonal (GO) initialization.

shows the normalized kernel functions evaluated at an ordered grid of points for a wide and deep neural network initialized using Glorot initialization, and the proposed GO initialization, respectively. Figure 4 empirically shows that MLPs initialized with the proposed GO scheme have a limiting

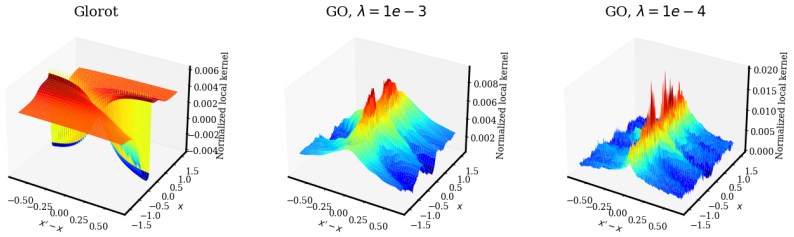


Figure 4: Normalized kernel functions in the domain for (from left to right) Glorot scheme, GO scheme with $\lambda = 10^{-3}$, GO scheme with $\lambda = 10^{-4}$. The x axis (facing outward to the left) is the distance between the center of the kernel, and y axis (facing outwards to the right), is the location of that the center of the kernel locates, and the z axis (facing upwards), shows the normalized value of the kernel function.

kernel function whose distance parameter is smaller, as the regularizer coefficient decreases. Smaller distance parameters suggest less correlation between the prediction and the data output whose input location is further away, thus preventing frequency bias arising from kernels with an overly large distance parameter that tend to average the oscillations brought by the high frequency components and only retain low frequency components in the prediction. Such tendency is also observed in previous works on the NTK of physics-informed neural networks (PINNs) [5, 8] in the context of solving multi-scale differential equations.

4 Results

Synthetic data: In this example we demonstrate the effectiveness of GO initialization, on a 256×16 MLP, in mitigating frequency bias and enabling the approximation of synthetic high frequency functions in two dimensions. Instead of increasing the frequency of the target function, we increase the size of the training data-set exponentially in each training session, until the training and testing errors converge. The training and testing errors associated with data-sets of various sizes are reported in Fig. 5. We in addition also test the orthonormal weights initialization, where the square weight matrices are the orthonormal matrices returned from the QR factorization of the originally initialized weight matrices, normalized to the original Frobenious norm. The MLPs with Glorot initialization fail to accommodate the fitting of increasingly complex target functions, and eventually collapse with a training and testing relative $\mathbb{L}_2$ errors of $\mathcal{O}(1)$. While the orthonormal initialization helps mitigate the collapsing problem, the large empirical confidence intervals still indicates that the MLP will still

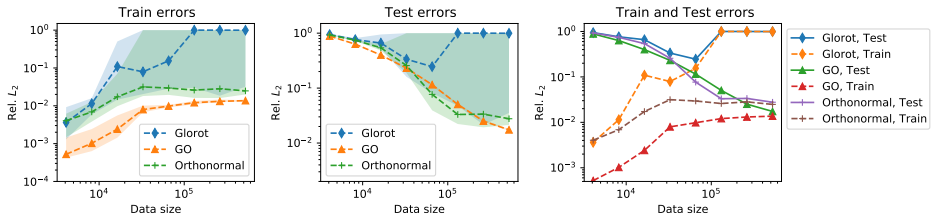


Figure 5: The median training errors (left), the median test errors (middle) on the same target function for different data sizes, and both errors in the same plot (right). The shaded region is the 80% intervals of 20 independent draws from the corresponding initialization schemes.

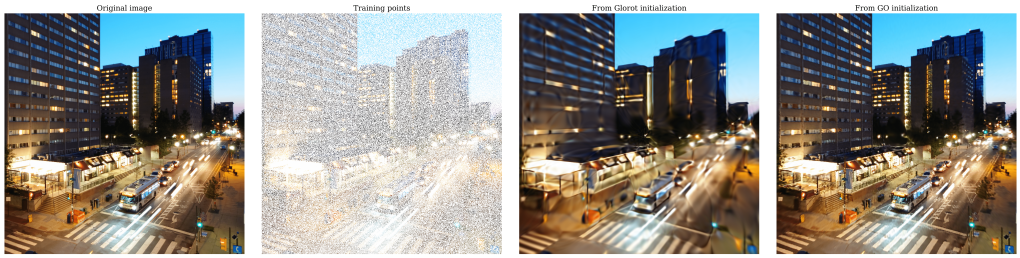


Figure 6: Training results. (From left to right) Original image, sampling mask, result from Glorot initialization, and result from the GO initialization.

suffer from that problem with considerable probability. On the contrary, the MLPs initialized with the GO scheme are able to capture broadband frequencies, and the empirical confidence interval of the errors also suggests that the MLP initialized with the GO scheme is less sensitive to the initial points, and is therefore more robust. However, we should acknowledge that the realization of the theoretical error bound and complexity of a given architecture could be heavily influenced by the choice of training hyperparameters, and our result in Fig. 5 shows that the initialization could be one of these crucial factors.

Image regression: In this example, we will use a high definition RGB colored image. The sample image contains rich details of a city landscape. A continuous representation of the image is recovered by fitting 25% of the pixels with total variation regularization. As shown in Fig. 6, an MLP with GO initialization is able to improve the SNR from 16.8 dB to 20.0 dB from the Glorot scheme, and recovers visually sharper details on light, shadow and texture that are on the contrary heavily distorted and smudged together in the representation trained from Glorot scheme. A comparison of SNRs for different sampling ratios and different competing approaches can be found in Appendix C.5.

5 Summary

By deriving a simplified discrete dynamical system to quantify information flow through MLPs at initialization, we identify and analyze vanishing output pathologies of conventional initialization schemes that hamper the trainability of deep fully-connected architectures. Motivated by these observations, we propose a novel initialization procedure that is designed to promote global orthogonality in the last layer's features, and demonstrate that such initialization leads to rich and localized feature representations. Our numerical results confirm that the proposed GO initialization can greatly enhance the trainability of deep MLPs and effectively mitigate frequency bias, without requiring any elaborate changes to the network architecture or activation functions.

Our findings motivate further investigation to elucidate the effect of orthogonal features as a mechanism to promote expressiveness of MLPs. In future work we plan to extend our analysis to consider a wider range of activation functions, as well as loss objectives for different problems. For example, for approximating solution of partial differential equations, the orthogonality objective may be augmented by the orthogonality of a set features that also shows diverse behavior after being transformed by a differential operator acting on the network's outputs.

Acknowledgements

This work received support from DOE grant DE-SC0019116, AFOSR grant FA9550-20-1-0060, and DOE-ARPA grant DE-AR0001201.

References

[1] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 09–15 Jun 2019.

[2] Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning, 2020.

[3] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies, 2019.

[4] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[5] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[8] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks, 2020.

[9] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661*, 2020.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Santiago, Chile, December 2015. IEEE.

[12] Eric C. Cyr, Mamikon A. Gulian, Ravi G. Patel, Mauro Perego, and Nathaniel A. Trask. Robust training and initialization of deep neural networks: An adaptive basis viewpoint, 2019.

[13] Gurpreet Singh, Soumyajit Gupta, and Clint N. Dawson. Prevention is better than cure: Handling basis collapse and transparency in dense networks, 2020.

[14] Richard Durrett. *Probability: theory and examples*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, Cambridge ; New York, 4th ed edition, 2010. OCLC: ocn607573997.

[15] Claudio Canuto, M Yousuff Hussaini, Alfio Quarteroni, and Thomas A Zang. *Spectral methods: fundamentals in single domains*. Springer Science & Business Media, 2007.

[16] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014.

[17] Wei Huang, Weitao Du, and Richard Yi Da Xu. On the neural tangent kernel of deep networks with orthogonal initialization, 2021.

[18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2020.

[19] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[20] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124002, Dec 2020.

[21] Amnon Geifman, Abhay Yadav, Yoni Kasten, Meirav Galun, David Jacobs, and Basri Ronen. On the similarity between the laplace and neural tangent kernels. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1451–1461. Curran Associates, Inc., 2020.

[22] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[23] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [N/A] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The experiments remain in toy level and the computational cost is much below average. We only include the computation cost for a few representative ones.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Computational cost

GO initialization is typically carried out using a relatively small number of stochastic gradient descent iterations. In all experiments reported in this work we employed $5,000$ iterations to roughly calibrate the network's weights and biases at initialization. This typically amounts to a fraction of the total cost required for training the network to fit observed data. For example, applying GO initialization to an MLP with 4 hidden layers and 32 neurons per layer using a batch size of 256 takes roughly 7 seconds, while for a larger MLP with 16 hidden layers and 256 neurons per layer, the initialization takes roughly 1 minute for a larger batch size of $2,048$ (all timings reported on a signle NVIDIA RTX A6000 GPU). We therefore consider the computational allocated to applying the GO initialization as negligible, compared to the cost of training. Furthermore, the initialization cost could be further reduced by fixing one draw of the parameters instead of generating different random draws of weights and biases at each iteration of optimizing the orthogonality objective.

B Back-propagation as a dynamical system

We analyze the backward pass of an MLP [22] as a dynamical system to show that the increased variance offset discussed in section 2 will not lead to vanishing gradients in MLPs activated with hyperbolic tangent activations. To this end, a linear layer followed by a hyperbolic tangent activation can be expressed as $Y = \tanh(XW)$, where $X \in \mathbb{R}^{N \times m_0}$ is the input of the hidden layer, and $W \in \mathbb{R}^{m_0 \times m_1}$ is the weight matrix. Let $dY = \frac{\partial l}{\partial Y}$, i.e., the back propagated gradient with respect to the objective l. We again assume that the entries of dY is distributed i.i.d. Then the gradient of the weight matrix is

$$dW = \frac{\partial l}{\partial W} = X^\intercal \tanh'(dY), \tag{B.1}$$

and the further back-propagated gradient is

$$dX = \frac{\partial l}{\partial X} = \tanh'(dY) W^\intercal. \tag{B.2}$$

To study the impact of the variance offset c on the back-propagated gradients dX, we choose to trace the square root of second moment of $T_X = \tanh'(dX)$. Let $s_X^T = \sqrt{\mathbb{E}(T_X^2)}$, $T_Y = \tanh'(T_X W^\intercal)$, and $s_Y^T = \sqrt{\mathbb{E}(T_Y^2)}$. Then, similarly, as $m_0 = m_1 \to \infty$, with the offset c, the distribution of dX is approximately normal with zero mean and standard deviation $c s_X^T$. Therefore, we can write a discrete dynamical system for back-propagated gradients as

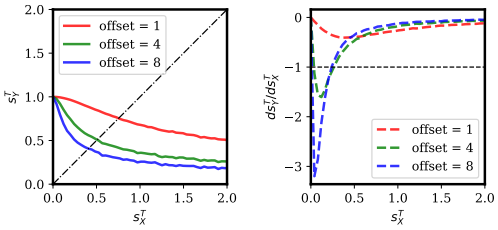
$$s_X^T = \sqrt{\mathbb{E}(\tanh'(c s_Y^T \mathcal{N}(0,1))^2)}. \tag{B.3}$$

Again, considering the difficulty in obtaining the analytic form of the integration, we resort to the law of large numbers to estimate the expectation. Furthermore, given that $\tanh'(x) = 1 - \tanh(x)^2$ which is an even function with monotonically decreasing component on $\mathbb{R}_+$, we have that

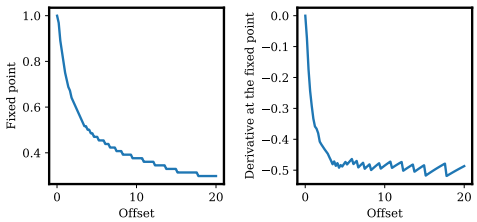
$$\tanh'(\sigma_{dX}^0 x)^2 < \tanh'(\sigma_{dX}^1 x)^2, \tag{B.4}$$

for all fixed x, if $\sigma_{dX}^0 > \sigma_{dX}^1 \geq 0$. Then since $\tanh'(x)$ is bounded (and therefore integrable), the right hand side of (B.3) decreases monotonically for positive s_Y^T. And since $\sqrt{\mathbb{E}(\tanh'(0\mathcal{N}(0,1))^2)} = 1$, the only fixed point of the system is positive. Fig. 7 shows the evolution of (B.3) at different variance offset value cs.

As before, we use a bisection method to roughly identify the fixed points given different variance offsets, and observe their stability. As shown in Fig. 7(b), our numerical results suggest that for a fixed variance offset at reasonable range, the only positive fixed point for the back-propagated gradient is stable. Moreover, although the second moment shrinks for larger variance offset, it does not vanish.



(a)



(b)

Figure 7: (a) The fixed point of Eq. (B.3) (left) and the its derivative at the root (right) for different variance offset values. (b) Positive fixed point (left), and its derivatives (right) of Eq. (B.3) for varying variance offset.

C Experimental setup

All numerical experiments presented in this work were performed in Python using the JAX library [23]. Unless otherwise specified, the optimizer is Adam with default settings [24]. The input and output of the data are normalized to zero mean and unit variance. In the following we provide the necessary details for reproducing the main results and conclusions presented in the manuscript.

C.1 Variance flow as a dynamical system

For Fig. 1(a,b), Eq. (2.1) and its derivative are approximated with 4096 draws from the unit normal distribution. For Fig. 1(c), the neural network consists of 32 layers, each with 1024 neurons, and a hyperbolic tangent activation function. Moreover, the function and its derivative shown in Fig. 2 are evaluated the same as that of Fig. 1. The fixed point is obtained by bisection method.

C.2 Globally Orthogonal (GO) initialization of MLPs

For the visualization of the function behavior at initialization depicted in Fig. 3, we choose 2^{16} data points selected uniformly from the square region $[-1, 1] \times [-1, 1]$ and apply normalization for both dimension. The width of the MLP network is set to be 256. The visualization of the outputs is on a 200×200 grid. The GO initialization scheme was applied using the Adam optimizer with a fixed learning rate of 0.005. The orthogonality loss objective is regularized with $\lambda = 10^{-4}$, and $10,000$ stochastic gradient descent steps were performed.

The results presented in Fig.4 are obtained using an MLP network with 16 layers, each with 256 neurons activated with the hyperbolic tangent function. The GO initialization is trained on 2048 of uniformly distributed points with zero mean and unit variance. The regularizer $\lambda = 10^{-3}$ and the learning rate is fixed at 0.005. The GO initialization is trained for $10,000$ iterations. The NTK is calculated on 256 evenly spaced points with zero mean and unit variance.

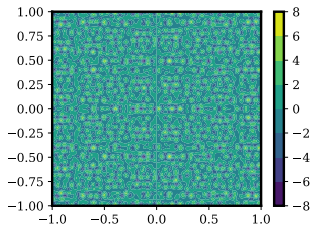


Figure 8: Target function used for the synthetic benchmark in section 4.

C.3 Synthetic benchmark

The results presented in Fig.5 correspond to a regression task using target function with the following form

$$\begin{aligned} f(x,y) &= \sum_{i=1}^{4} A_{i,0}\sin(2\pi\nu_{i,0}x)\cos(2\pi\nu_{i,1}y) + A_{i,1}\sin(2\pi\nu_{i,2}x)\sin(2\pi\nu_{i,3}y), \\ \nu_{i,j} &\sim U(0,M), i=1,...,4, j=0,1,2,3 \\ A_{i,j} &\sim U(0,2), i=1,...,4, j=0,1. \end{aligned} \tag{C.1}$$

where $M = 18$ is the maximum frequency. Fig. 8 shows the target function that is used for data generation.

Here we have considered an MLP network with 16 layers, each with 256 neurons activated by the hyperbolic tangent. For the GO initialization, 2^{16} data points are generated uniformly randomly in the square $[-1,1]\times[-1,1]$. They are then normalized in each dimension. The GO initialization is trained with $\lambda = 10^{-4}$ and a learning rate of 0.005. For each iteration, $2,048$ data points are selected uniformly from the pre-drawn points. The initialization is trained for $20,000$ iterations.

Once the model is initialized with proceed with solving the regression task, and 2048 data points are chosen per batch uniformly and randomly from the data-set with given size. The MLP is then trained using the Adam optimizer with a learning rate of $5e-4$ and a decay rate of 0.99 every 400 iterations, for a total of $\sim 100,000$ iterations. We have performed a total of 20 independent trials with parameters initialized from both schemes (i.e. Glorot and GO), and use the ensemble to estimate the 80% probability interval. The testing error is computed on 200×200 regular grid, normalized by the training data normalization.

C.4 Image regression benchmark

The benchmark presented in Fig.6 involves a real RGB image with a resolution of $960\times 960\times 3$. We use 25% of the pixels as the training data-set, and the signal-to-noise ratio (SNR) is evaluated on the entire image including the training data. The MLP network we employed consists of 8 layers with a width of 512, and a hyperbolic tangent activation function. The GO initialization is trained using the same exact settings is in the synthetic example discussed above, using a batch size of 512.

Once the model is initialized we proceed to solve the regression task, using the MSE loss objective and a total variation (TV) regularization on the Frobenius norm of the MLP Jacobian matrix

$$L(w|(X^1,Y^1),X^2) = \frac{1}{N_1}\sum_{j=1}^{N_1}||f_w(X_j^1)-Y_j^1||_2^2 + \frac{\lambda_r}{N_2}\sum_{i=1}^{N_2}||J_{f_w}(X_i^2)||_2^2, \tag{C.2}$$

where X^1 is a batch of normalized indices, Y^1 is a batch of corresponding RGB value vectors, X^2 is a batch of points for evaluating the total variation regularization, λ_r is the regularization coefficient, and N_1, N_2 are the corresponding batch sizes. In this case X^1 is drawn uniformly randomly from the sampled normalized indices, X^2 is sampled uniformly randomly in the normalized domain, $N_1 = N_2 = 512$, and $\lambda_r = 0.01$. The number of training iterations is $400,000$. The optimizer learning rate starts from 10^{-4}, and decays with a rate of 0.99 per $1,000$ steps.

Sample ratio	Init. Scheme	$\lambda_r = 0$	$\lambda_r = 0.01$	$\lambda_r = 1$
10%	MLP, Glorot	16.1	16.1	16.1
	MLP, GO	16.8	17.0	17.4
	SIREN, $\omega_0 = 1$	16.25	16.44	17.46
	SIREN, $\omega_0 = 30$	16.11	17.4	17.19
	SIREN, GO	16.42	16.72	17.50
25%	MLP, Glorot	16.7	16.8	16.7
	MLP, GO	19.9	20.0	19.7
	SIREN, $\omega_0 = 1$	19.45	19.47	19.27
	SIREN, $\omega_0 = 30$	19.46	20.32	19.88
	SIREN, GO	19.79	19.85	19.89
99%	MLP, Glorot	16.9	16.9	16.9
	MLP, GO	22.4	22.4	21.0
	SIREN, $\omega_0 = 1$	20.94	20.86	20.01
	SIREN, $\omega_0 = 30$	32.81	32.66	23.23
	SIREN, GO	23.15	23.07	21.33

Table 1: SNRs in decibel of different initialization schemes with sample ratio $0.1, 0.25, 0.99$ and total variation regularization coefficient $\lambda_r = 0, 0.01, 1$ on the colored landscape image. SIREN is sine activated MLP with the recommended initialization scheme from the original paper [9], except for ω_0, which is the offset for the first weight matrix. The original paper uses $\omega_0 = 30$, however, we also provides the case where $\omega_0 = 1$ for readers' reference. The impact of GO on the SIREN is not as evident as that on the MLP with tanh activation, and it has been recognised by Tancik *et al.*[7] that the Fourier features frequency, i.e., the additional offset by the first layer, is hard to be updated with certain first order gradient based optimization. However, the GO scheme is able to marginally improve the performance of SIREN when data is sufficient.

C.5 Comparison against SIREN

For a comparison against state-of-the-art methods for image regression, we report the results obtained by an MLP with GO initialization versus a scaled SIREN model [9]. The weight matrices of the SIREN network are initialized in accordance with [9], without using any heuristics on the first layer variance offset, and the architecture stays the same, except for the periodic activation function used in the SIREN case. Table 1 shows the SNRs for different configurations. Interestingly, the proposed GO initialization can achieve comparable image reconstructions to the SIREN network, suggesting that the proposed GO initialization alone can be sufficient in mitigating frequency bias, without any additional modifications to the network architecture or activation functions.