Training Dynamics of Convolutional Neural Networks for Learning the Derivative Operator

Erik Y. Wang[1,3], **Yongji Wang**[2,3], **Ching-Yao Lai**[3]
[1]Harvard University, [2]New York University, [3]Stanford University
erikwang@college.harvard.edu, yw8211@nyu.edu, cylai@stanford.edu

Abstract

Despite significant interest in developing new methods for scientific machine learning, the behavior and effectiveness of existing methods are not thoroughly understood. For instance, while deep multi-layer perceptrons are known to exhibit a bias toward low-frequency data, whether this phenomenon holds for other methods and how it manifests are less certain. We investigate the training dynamics of convolutional neural networks in the context of operator learning and find that the input signal's high-frequency components are generally learned before its low-frequency components, followed by the amplitudes of the frequency distribution. Our results also show that networks trained on a range of frequencies tend to perform better on high-frequency data. From an architectural standpoint, increasing the model's kernel size can decrease this accuracy gap and improve precision across frequencies, but this trend does not hold for deeper models, suggesting that larger kernel sizes may have a stronger effect on training stability and model accuracy.

1 Introduction

Neural networks have emerged as a powerful tool for solving scientific problems, particularly in learning mappings between function spaces. While traditional numerical methods can solve differential equations with high accuracy, they often face computational limitations and lack flexibility. For instance, solving a complex partial differential equation (PDE) yields a solution for only one specific case and can be computationally expensive. In contrast, neural network-based approaches can efficiently learn and provide solutions for entire families of functions [7, 12, 15, 17].

Despite their success, neural networks—their training dynamics and spectral behavior across data in particular—remain poorly understood. This knowledge gap limits potential improvements in model robustness and accuracy, especially in complex, multiscale applications like fluid dynamics or climate modeling. For instance, a key challenge in deep neural networks is the spectral bias problem, where networks learn high-frequency components before low-frequency ones. This can be especially problematic in complex, multiscale settings with small-scale features—such as those encountered in fluid dynamics or climate modeling [3, 9, 14, 16, 18, 23]. Despite having been studied in multi-layer perceptrons (MLPs), the spectral bias is not well-established in other important architectures such as convolutional neural networks (CNNs) and neural operators.

In this work, we study the training dynamics of CNNs in a regression setting. We are interested in understanding (i) what the CNN learns during training and (ii) whether certain architectural design choices can improve the model's performance. In particular, we aim to understand how the depth and kernel size of a CNN influence its performance and robustness to different frequencies. To that end, we generate datasets of one-variable functions as well as their derivatives and train CNNs to learn the derivative operator $\frac{d}{dx}$. Our work builds on [20, 27] but extends it to operator learning and the CNN architecture, and also specifically studies the early training dynamics of CNNs.

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

2 Related work

Several works have shown from a Fourier perspective that neural networks often initially learn low-frequency functions [20, 27–29]. Xu et al. [29] specifically show evidence of larger error on high-frequency data in CNNs, but this work uses image datasets—MNIST and CIFAR—instead of one-dimensional symbolic functions in our setting. Neural Tangent Kernel (NTK) theory has also been used to demonstrate that in very wide networks, components within the NTK's top eigenspace are learned more quickly than others [2, 8, 22]. Particularly relevant to our study is the finding that during the early training stages, a neural network's behavior can largely be approximated by a linear function of the data [10].

There has also been some work in designing neural networks that can better learn high-frequency data [24, 26]. In the context of scientific machine learning, spectral bias refers to the tendency of deep networks to learn lower-frequency components of the data before higher-frequency components, but research in computer vision has shown a distinct yet related phenomenon in CNNs, wherein high-frequency information plays a critical role in classification accuracy [5, 25]. In particular, Abello et al. [1] found that middle or high-level frequencies have a disproportionately large impact on model robustness in CNNs. In other words, CNNs can retain their accuracy when provided with only high-frequency components of an image but struggle when given only low-frequency components. In this study, we focus on the behavior of CNNs for regression tasks.

3 Methodology

3.1 Data preparation

We prepare three datasets of functions $u_g(x)$ that have distinct ranges of frequencies: the "low," "general," and "high-frequency" datasets have frequencies between 1 and 5, 1 and 10, and 6 and 10, respectively. For each dataset, the functions generated have the form

$$u_g(x) = \sum_{i=1}^{N} A_i \sin(F_i x + \Phi_i) + \sum_{i=1}^{N} B_i \cos(F_i x + \Psi_i), \tag{1}$$

where $N = 10$, A_i and B_i are random integers between 0 and 1 that set the amplitude of each term, F_i is a random integer between the minimum and maximum allowed frequencies, and Ψ_i are random floats between 0 and 2π specifying the phase shift for each term. Each dataset contains 500 randomly generated functions, each with 1000 uniform samples from $x \in [0, 2\pi]$. The training data is generated using PyTorch, which is assumed to provide numerically precise derivatives via its automatic differentiation engine [19]. The frequency spectrum of each function is computed using the one-dimension Discrete Fourier Transform (DFT), given by

$$\hat{f}(k) = \frac{1}{N} \sum_{n=0}^{N-1} f(x_n) \exp(-ikx_n), \tag{2}$$

where N is the number of equally spaced points in the domain, and k is the frequency [6]. Fig. 1a shows a random function from the dataset along with its ground-truth derivative, and Figs. 1b and 1c visualize the frequencies of the function using a Fourier transform.

3.2 Training details

We consider the supervised learning regression framework where $N_d = 1000$ data points are given: $\{(u_g(x_i), u_g'(x_i))\}_{i=1}^{N_d}$. Here, $\mathbf{u_g} : \mathbb{R} \to \mathbb{R}$ represents a function discretized at points $\{x_i\}_{i=1}^{N_d}$ in its domain, and $\mathbf{u_g'}$ is its derivative. In other words, the discretizations are given by:

$$\mathbf{u_g} = [u_g(x_1), u_g(x_2), \ldots, u_g(x_{N_d})],\ \mathbf{u_g'} = [u_g'(x_1), u_g'(x_2), \ldots, u_g'(x_{N_d})].$$

The objective is to train a neural network f that maps the discretized function values $\mathbf{u_g}$ to the corresponding derivative values $\mathbf{u_g'}$. Models are trained using the standard mean-squared error (MSE) but evaluated using the normalized mean-squared error (NMSE) to account for the higher variability in high-frequency functions, which often results in larger losses. We train CNNs with varying layers

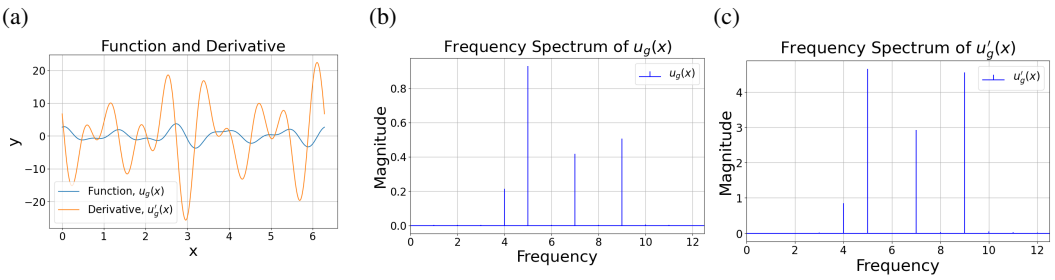

Figure 1: **Dataset information.** (a) Visualizing a random function sampled from the dataset and its ground-truth derivative. (b) Frequency information of the function in (a). (c) Frequency information of the ground-truth derivative.

and kernel sizes, but with each not less than 3. Each hidden layer has 64 input and output channels, with padding set to half the kernel size to preserve spatial dimensions, using a stride of 1. All models are trained on the general frequency dataset.

4 Experiments

4.1 Early training dynamics

Analyzing the frequency of the model's outputs during training provides insights into how and what the model learns. First, we find that it is necessary to decrease the learning rate of the optimization from 10^{-3} to 10^{-4} to observe any changes visually in the frequencies of the model's outputs between epochs, as it was difficult to observe the learning process visually with a learning rate higher than 10^{-3}. Fig. 2e shows that within 10 epochs of training, the model learns the high-frequency components of the derivative before the low-frequency components, indicated by high-frequency components having larger magnitudes, but the correct distribution of dominant frequencies is quickly learned by the model (as indicated by Fig. 2f, which shows the frequency heatmap at epoch 100). Since a log scale is used for the magnitude colormap, the frequency distribution looks largely correct without needing a very large number of training epochs.

Even though the CNN learns the frequency distribution quickly, the amplitudes are incorrect. For instance, at epoch 100, the frequency heatmap looks comparable to that of the ground-truth derivative frequency heatmap (Fig. 2g), but the loss remains high as the amplitudes are incorrect relative to the true amplitudes. As training progresses, the amplitudes gradually increase to their correct values as defined by the ground-truth derivative data, suggesting that the model quickly learns information about the frequencies of the solution and gradually learns the amplitudes of the frequencies later.

4.2 Effect of larger kernel sizes

With a fixed number of layers, increasing the kernel size tends to increase the speed at which the model learns the distribution of frequencies. For a CNN with three layers and a kernel size of three, changes in the frequency distribution of the model's outputs can be visually seen within the first ten epochs of training using a learning rate of 10^{-4}. However, to see similar changes (i.e., Fig. 2) in a model with a kernel size of 31, it was necessary to decrease the learning rate to 10^{-6}. The training dynamics were extremely similar visually: high-frequency information was learned before low-frequency information, followed by the magnitudes for the frequencies. Since a smaller learning rate was required to see comparable spectral behavior to the model using a smaller kernel size, increasing the kernel size may have a powerful effect on improving the model's ability to learn the data's spectral information. This effect can be attributed to expanding the model's receptive window, similar to how increasing the order of a numerical method can decrease its truncation error by providing the model with more spatial information. However, we show in Appendix A.4 that increasing the number of layers in the model does not provide similar improvements in performance: deeper networks decrease model accuracy and training stability, which can likely be explained by the curse of dimensionality [4].

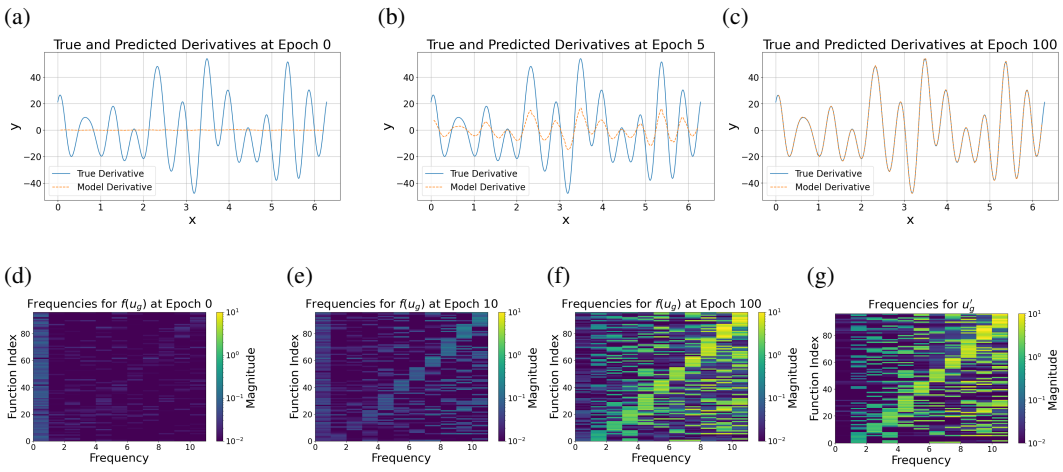

Figure 2: **Early training dynamics.** (a–c) A model quickly learns frequency information before the amplitudes. (d–g) shows heatmaps of the general frequency dataset dominant frequencies. In (e–f), one can see a marginally brighter magnitude in the top right corner with increasing iterations, which corresponds to larger amplitudes at higher amplitudes. By 100 epochs, the distribution of the spectral information is nearly indistinguishable from the ground-truth dataset, shown in (g).

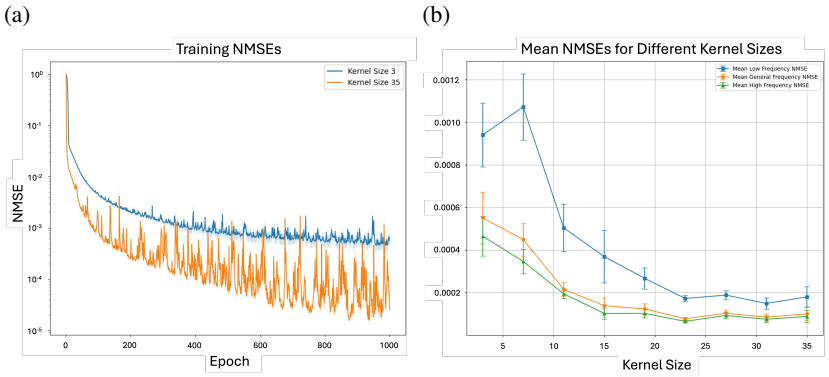

Figure 3: **Effects of different kernel sizes.** (a) compares the mean NMSE of models with the smallest (3) and largest (35) kernel sizes over five random seeds. (b) visualizes the effect of larger kernel sizes on model accuracy over different datasets. NMSEs are averaged over the last 50 epochs.

We also find that models with larger kernel sizes can be trained to higher precision than models with smaller kernel sizes (using the NMSE to measure the accuracy of a model). In other words, models with larger kernel sizes reach a lower NMSE plateau after training. Evaluations of models on the low-frequency and high-frequency function datasets show a higher NMSE on the low-frequency datasets than on the high-frequency datasets, which is similar to the inverse of the spectral bias problem in MLPs, in which low-frequency functions are expected to be learned more accurately than high-frequency functions. This trend holds throughout the entire training process but weakens with larger kernel sizes. The accuracy gap between the low and high-frequency datasets is also the largest during the early stages of training and gradually decreases over iterations, but still exists after 2000 epochs, when the NMSE reaches roughly 10^{-4} to 10^{-5}.

5 Discussion

We investigate the training dynamics of CNNs from a spectral perspective by focusing on the derivative operator. During early stages of training, models appear to rapidly learn the frequency information of the solution operator while prioritizing high-frequency components over low-frequency components

of the data. This observation of high-frequency information being preferentially learned contrasts with the typical notion of spectral bias that has been shown to exist in MLPs, where networks fit functions from low to high frequency. However, this phenomenon may occur because the derivative operator scales the amplitude of a function by its frequency. Since derivatives of higher-frequency functions therefore have larger amplitudes compared to those of lower-frequency functions, the loss function will place more weight on the higher-frequency functions.

Certain architectural choices also seem to have a significant impact on model precision and robustness. Increasing the kernel size while keeping the number of layers constant accelerates the model's ability to learn spectral information, whereas increasing the number of layers while fixing the kernel size does not yield the same effect. Larger kernel sizes also help narrow the accuracy gap between low-frequency and high-frequency datasets. Finally, we show that using NMSE as a loss function instead of MSE can improve the stability of the model training process on continuous data with a wide range of frequencies.

While our results are indicative of a form of bias, future work will focus on scaling the amplitudes of lower-frequency functions to account for the larger amplitudes of high-frequency derivatives and enable a more thorough analysis of whether a spectral bias indeed exists. We will also expand our analysis to more challenging settings, such as learning the solution to a family of PDEs or leveraging prior information about the system [13, 21]. Finally, we plan to explore whether regularization techniques can help improve training stability or generalization across frequency regimes. For instance, Yin et al. [30] found that low-frequency data improves robustness to high-frequency corruptions like noise and blur but degrades performance on low-frequency corruptions like fog and contrast. Therefore, we plan to experiment with adding mixtures of noise to our data to determine whether the spectral bias can be mitigated by adding different frequencies of data to the training process.

References

[1] Antonio A Abello, Roberto Hirata, and Zhangyang Wang. Dissecting the high-frequency bias in convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 863–871, 2021.

[2] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[3] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

[4] Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

[5] Rémi Bernhard, Pierre-Alain Moëllic, Martial Mermillod, Yannick Bourrier, Romain Cohendet, Miguel Solinas, and Marina Reyboz. Impact of spatial frequency based constraints on adversarial robustness. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[6] Ronald N Bracewell. The fourier transform. *Scientific American*, 260(6):86–95, 1989.

[7] Steven L Brunton and J Nathan Kutz. Promising directions of machine learning for partial differential equations. *Nature Computational Science*, pages 1–12, 2024.

[8] Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2021.

[9] Ashesh Chattopadhyay and Pedram Hassanzadeh. Long-term instabilities of deep learning-based digital twins of the climate system: The cause and a solution. *arXiv preprint arXiv:2304.07029*, 2023.

[10] Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `https://api.semanticscholar.org/CorpusID:6628106`.

[12] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[13] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

[14] Ching-Yao Lai, Pedram Hassanzadeh, Aditi Sheshadri, Maike Sonnewald, Raffaele Ferrari, and Venkatramani Balaji. Machine learning for climate physics and simulations. *arXiv preprint arXiv:2404.13227*, 2024.

[15] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[16] Xinliang Liu, Bo Xu, Shuhao Cao, and Lei Zhang. Mitigating spectral bias for the multiscale operator learning. *Journal of Computational Physics*, 506:112944, 2024.

[17] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[18] Rambod Mojgani, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Interpretable structural model error discovery from sparse assimilation increments using spectral bias-reduced neural networks: A quasi-geostrophic turbulence test case. *Journal of Advances in Modeling Earth Systems*, 16(3):e2023MS004033, 2024.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[20] 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 *International conference on machine learning*, pages 5301–5310. PMLR, 2019.

[21] 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.

[22] Lili Su and Pengkun Yang. On learning over-parameterized neural networks: A functional approximation perspective. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Adam Subel, Yifei Guan, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Explaining the physics of transfer learning in data-driven turbulence modeling. *PNAS nexus*, 2(3):pgad015, 2023.

[24] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

[25] Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P Xing. High-frequency component helps explain the generalization of convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8684–8694, 2020.

[26] 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. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.

[27] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

[28] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part I 26*, pages 264–274. Springer, 2019.

[29] Zhi-Qin John Xu, Yaoyu Zhang, and Tao Luo. Overview frequency principle/spectral bias in deep learning. *Communications on Applied Mathematics and Computation*, pages 1–38, 2024.

[30] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. *Advances in Neural Information Processing Systems*, 32, 2019.

A Appendix

A.1 Additional training details

The activation layer between layers in each CNN is ReLU, except for the final convolutional layer, which directly outputs the convolved feature without an activation function given our continuous regression setting. Models are trained with mini-batch gradient descent using the Adam optimizer [11] and a learning rate of 10^{-3} on the general frequency dataset, which contains functions with frequencies between one and ten. Each dataset is split into 400 functions for training and 100 functions for evaluation. All models are trained on a single GPU.

A.2 Frequency heatmaps

To visually inspect the training dynamics of the model, heatmaps of the data in frequency space are constructed. Each heatmap displays the original functions u_g from the general frequency dataset in frequency space. Function indices are on the y-axis, where functions are sorted in ascending order by their dominant frequency (the frequency component with the largest magnitude), each function's frequency components are on the x-axis, and the magnitudes of the frequencies are arranged on the colorbar in logarithmic scale.

A.3 Visualization of the learning process

Given an arbitrary function, the model produces increasingly accurate outputs over iterations. Notably, the model requires very little time to correctly learn the shape of the output, even though the magnitudes of the function are incorrect. This is consistent with the frequency heatmaps shown in Figs. 2 d–f, which shows the spectral behavior of the model's outputs during training.

A.4 NMSE as the loss

Training models using the NMSE instead of the MSE as the loss function creates a considerably different training loss trajectory. In particular, the loss (NMSE) quickly plateaus at early stages during training, with several small decreases before following a more typical trajectory after a certain number of epochs. In Fig. 4a, after around 500 epochs, the optimization finds a new loss minima for the model, after which a rapid decrease in NMSE is observed. Additionally, model training is more stable using the NMSE as the loss function; Fig. 4b shows that the model trained using the MSE experiences larger oscillations in NMSE across the low-frequency and high-frequency datasets. However, the accuracy gap between datasets is visible during the entire training process in both setups, suggesting that even with a normalized error for training, the model may be biased to preferentially learn high-frequency data during training.

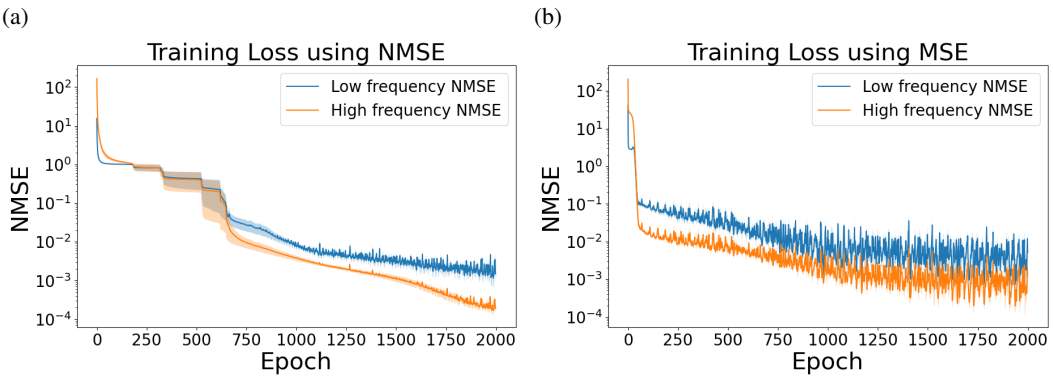

Figure 4: Comparison of the loss curves between a model trained using NMSE (left) and a model trained using MSE (right).

A.5 Effect of increased layers

Fig. 3 shows that larger kernels can substantially improve the accuracy of the model while reducing the performance gap between datasets of functions with different frequency ranges. Models with a larger kernel size also show smaller variances in NMSE across different training runs. We find that conversely, increasing the depth of the model tends to degrade performance and decrease the stability of training, which is consistent with an overparameterized network. The accuracy difference across datasets of different frequencies appears to widen

with additional layers, though the trend is less consistent compared to the steady improvement in NMSE with larger kernel sizes. As such, there is no clear benefit to increasing model depth in this setting. The formula used for the first-order forward difference method indicated by the horizontal line in Fig. 6 is given by

$$f'(x) = \frac{f(x + dx) - f(x)}{dx}, \tag{3}$$

where $f(x) = u_g(x)$ and $dx = 10^{-3}$ in our setting.

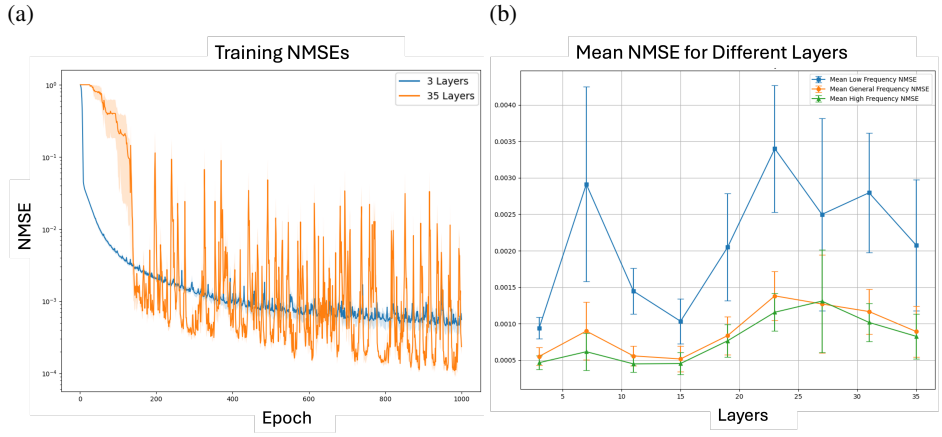

Figure 5: **Effect of increased model depth.** (a) compares the mean NMSE of models with the fewest and most number of layers. (b) visualizes the effect of increasing model size on model accuracy on different datasets. Both plots are averaged over five random seeds.

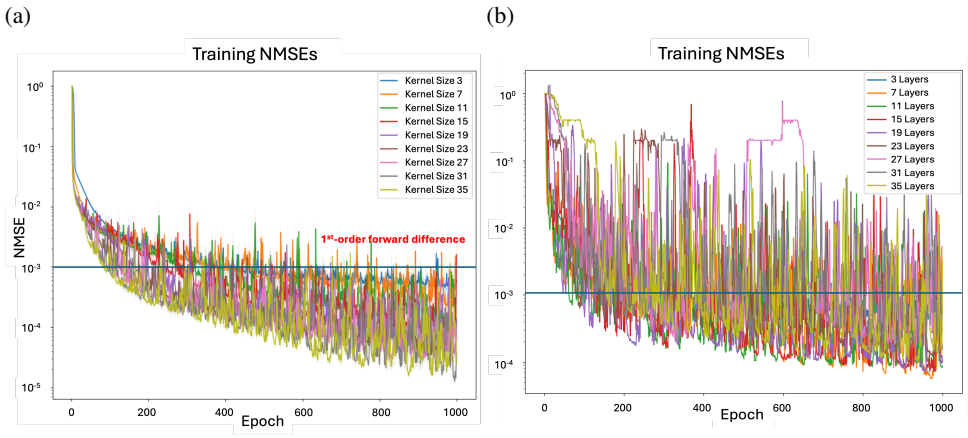

Figure 6: **Effect of training with larger kernel sizes versus more layers.** (a) shows the mean NMSE loss using different kernel sizes, while (b) uses the different model depths. Both plots are averaged over five different random seeds.