
Frequency Principle in Deep Learning with General Loss Functions and Its Potential Application

Zhi-Qin John Xu*

New York University Abu Dhabi
Abu Dhabi 129188, United Arab Emirates
zhiqinxu@nyu.edu

Abstract

Previous studies have shown that deep neural networks (DNNs) with common settings often capture target functions from low to high frequency, which is called Frequency Principle (F-Principle). It has also been shown that F-Principle can provide an understanding to the often observed good generalization ability of DNNs. However, previous studies focused on the loss function of mean square error, while various loss functions are used in practice. In this work, we show that the F-Principle holds for a general loss function (e.g., mean square error, cross entropy, etc.). In addition, DNN's F-Principle may be applied to develop numerical schemes for solving various problems which would benefit from a fast converging of low frequency. As an example of the potential usage of F-Principle, we apply DNN in solving differential equations, in which conventional methods (e.g., Jacobi method) is usually slow in solving problems due to the convergence from high to low frequency.

1 Introduction

Deep neural networks (DNNs) has achieved many state-of-the-art results in various fields [1], such as object recognition, language translation and game-play. A fully understanding of why DNNs can achieve such good results remains elusive. The often used DNNs equip much more parameters than the number of the training data. As Von Neumann said "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk". It is no surprise that such DNNs can well fit the training data. However, counter-intuitive to the traditional learning theory, such DNNs often do not overfit (DNNs often generalize well to the test data which are not seen during the training), which is often referred to as "apparent paradox" [2].

A series of recent works [3, 4, 5, 6, 7, 8, 9, 10], both experiments and theories, have gained us more understanding to this paradox. In this work, we focus on the Fourier analysis of DNNs [3, 4, 5]. Although the often used dataset, such as MNIST and CIFAR, are relative simple compared with practical dataset, the input dimension (the pixel number of each input image) is still very high for a quantitative analysis. A good starting point to understand this apparent paradox is to find an example that is simple enough for analysis but also preserves this interesting paradox. The simple example turns out to be the fitting of a function with one-dimension (1-d) input and 1-d output [3]. A prompt example to understand the apparent paradox is that a very high-order polynomial fitting for randomly sampled data points often overfit the training data, that is, high oscillation occurs around the sample boundary (Runge's phenomenon); however, a DNN with small weight initialization, no matter how large size the DNN is, often learns the training data with a relative flat function [10]. Starting from such 1-d functions, experimentally [3, 5] and theoretically [4], there exists a *Frequency Principle*

*This work is done while Xu is a visiting member at Courant Institute of Mathematical Sciences, New York University, New York, United States.

(F-Principle) that DNNs often first quickly capture low-frequency components while keeping high-frequency ones small, and then relatively slowly captures high-frequency components. By F-Principle, the high-frequency components of the DNN output is controlled by the training data. High oscillation which exists in the Runge’s phenomenon is then absent in the DNN fitting. F-Principle also holds well in the often used dataset [3], that is, MNIST and CIFAR-10. Theoretical work indicates that the key ingredient underlying the F-Principle in general DNN fitting problems is that the power spectrum of the activation function decays in the Fourier space, where the power-decay property is easy to be satisfied, such as sigmoid function and rectified linear unit.

Previous studies focused on the DNN with mean square error [3, 4, 5]. It is yet to study whether F-Principle applies in the DNN with other types of loss functions. This is important since loss function varies in different problems, such as image classification and solving differential equations [11]. In this work, we perform a theoretical analysis to show that for a general loss function, e.g., cross entropy, the F-Principle qualitative holds in the DNN training, which is also verified by experiments. The first experiment is a classification problem with the loss function of cross entropy. The second experiment is to apply DNN to solve Poisson equation by using Dirichlet’s principle.

The DNN is a powerful tool to solve differential equations [11, 12, 13], especially for high-dimensional problems. It is well-known that different frequencies converge with different speeds in solving differential equations by numerical schemes. For example, for the Jacobi method, low frequency converges much slower than high frequency. Multigrid method is designed to speed up the convergence, which explicitly first captures low-frequency parts [14]. In addition, manual frequency marching from low frequency to high frequency has achieved great success in designing numerical schemes in various problems, such as inverse scattering problems [15] and Cryo-EM reconstruction problems [16]. By showing F-Principle in solving Poisson’s equations, we emphasize that the DNN structure, which implicitly endows low frequency with high priority, could be a powerful tool to the problems that benefit from a fast converging of low frequency. For example, we propose an ideal that combines DNN and conventional methods (e.g., Jacobi method or Gauss-Seidel method), in which DNN is in charge of capturing low-frequency parts and conventional methods are in charge of capturing high-frequency parts. This idea is exemplified by solving a 1-d Poisson’s equation.

2 F-Principle with general loss function

Consider a general DNN, and denote its output as $\Upsilon_\theta(x)$, where θ stands for the DNN parameters and x stands for the input. Represent $\Upsilon_\theta(x)$ with orthonormal basis $\{p_k(x)\}$:

$$\Upsilon_\theta(x) = \sum_k c_{\theta,k} p_k(x), \quad (1)$$

where $c_{\theta,k}$ is the coefficient of mode k depending on θ . Denote the loss at sample x as

$$l_x = l(\Upsilon_\theta(x)). \quad (2)$$

The total loss L is

$$L = \sum_x l_x = \sum_x l(\Upsilon_\theta(x)). \quad (3)$$

Consider the gradient of the total loss with respect to parameter θ :

$$\frac{\partial L}{\partial \theta} = \sum_x \frac{\partial l(\Upsilon_\theta(x))}{\partial \theta} \quad (4)$$

$$= \sum_x \frac{\partial l(\Upsilon_\theta(x))}{\partial \Upsilon} \frac{\partial \Upsilon_\theta(x)}{\partial \theta} \quad (5)$$

$$= \sum_x \frac{\partial l(\Upsilon_\theta(x))}{\partial \Upsilon} \sum_k p_k(x) \frac{\partial c_{\theta,k}}{\partial \theta} \quad (6)$$

$$= \sum_k \frac{\partial c_{\theta,k}}{\partial \theta} \sum_x \frac{\partial l(\Upsilon_\theta(x))}{\partial \Upsilon} p_k(x). \quad (7)$$

Let

$$d_k \triangleq \sum_x \frac{\partial l(\Upsilon_\theta(x))}{\partial \Upsilon} p_k(x). \quad (8)$$

d_k is the coefficient of $\partial l(\Upsilon_\theta(x))/\partial \Upsilon$ at the component of p_k . Consider that $\{p_k(x)\}$ is Fourier basis. According to Riemann-Lebesgue lemma, if a function is an integrable function on an interval, then the Fourier coefficients of this function tend to 0 as the order k tends to infinity. Therefore, when the activation function and the target function both are integrable functions on the considered interval, $c_{\theta,k}$ and d_k tend to 0 as the order k tends to infinity. Denote

$$\mathcal{L}_k \triangleq \sum_k \frac{\partial c_{\theta,k}}{\partial \theta} d_k. \quad (9)$$

We have

$$\frac{\partial L}{\partial \theta} = \sum_k \mathcal{L}_k. \quad (10)$$

Therefore, we can decompose $\partial L/\partial \theta$ into a summation of \mathcal{L}_k , which tends to 0 as the order k tends to infinity. This analysis implies that for any loss function, the change of any parameter θ at each training step is affected more by lower frequencies, which would rationalize the F-Principle in general loss functions, as examined in the following experiments.

3 Experiment: cross entropy loss

The loss function of cross entropy is widely used in classification problems. We use experiments to show that F-Principle holds in the DNN training with this loss function.

3.1 Toy data

Consider a target function $y(x) = (y_1(x), y_2(x))$, where

$$y_1(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}, \quad (11)$$

$$y_2(x) = \begin{cases} 0 & x > 0 \\ 1 & x \leq 0 \end{cases}. \quad (12)$$

This fitting problem is a toy classification problem. In the DNN in this problem, the output layer has two neurons with softmax as activation function. The output is denoted as $\Upsilon(x) = (\Upsilon_1(x), \Upsilon_2(x))$. The loss function is

$$L = \sum_{j=1}^2 L_j = - \sum_{j=1}^2 \sum_x y_j(x) \log \Upsilon_j(x) + (1 - y_j(x)) \log(1 - \Upsilon_j(x)). \quad (13)$$

For illustration, we focus on $y_1(x)$, which is shown in Fig.1a. Next, we examine the convergence of different frequencies. In a finite interval, the frequency components of a target function are quantified by Fourier coefficients computed from Discrete Fourier Transform (DFT). Note that because the frequency in DFT is discrete, we can refer to a frequency component by its index instead of its physical frequency. The Fourier coefficient of $y_1(x)$ for the γ -th frequency component is denoted by $F[y_1](\gamma)$ (a complex number in general). $|F[y_1](\gamma)|$ is the corresponding amplitude, where $|\cdot|$ denotes the absolute value. Note that we call γ the *frequency index*. $|F[y_1](\gamma)|$ is shown in Fig. 1b. To examine the convergence behavior of different frequency components during the training of a DNN, we compute the relative difference of the DNN output $\Upsilon_1(x)$ and $y_1(x)$ in frequency domain at each recording step, i.e.,

$$\Delta_F(\gamma) = \frac{|F[\Upsilon_1](\gamma) - F[y_1](\gamma)|}{|F[\Upsilon_1](\gamma)|}. \quad (14)$$

During the training, $\Upsilon_1(x)$ captures $y_1(x)$ from low to high frequency in a clear order, as shown in Fig.1c.

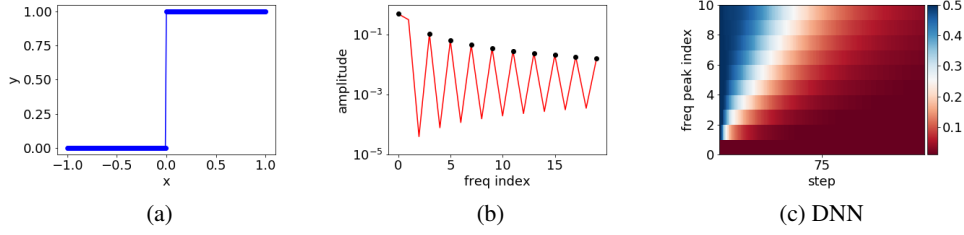


Figure 1: F-Principle with cross entropy loss. The first output dimension of the target function is shown in (a) and its Fourier coefficient amplitude as a function of frequency index is shown in (b). Frequency peaks are marked by black dots. (c) Relative difference at different recording steps for different selected frequency indexes. The training data are evenly sampled in $[-1, 1]$ with sample size 201. We use a DNN with width 400-400-200-100 with full batch training, the output layer has two neurons with softmax as activation function, and learning rate is 2×10^{-4} . The parameters of the DNN are initialized following a Gaussian distribution with mean 0 and standard deviation 0.1.

3.2 MNIST data

To verify that the F-Principle holds in the image classification problems (MNIST) with the loss function of cross entropy, we perform Fourier analysis in the first principle component of the input space. The procedure is as follows.

The training set is a list of images with labels: $\{\vec{x}_k, \vec{y}_k\}_{k=0}^{n-1}$. Each image is represented by a vector $\vec{x}_k \in \mathbb{R}^{N_{in}}$, where $N_{in} = 784$ is the pixel number of an image. \vec{y}_k is a one-hot vector indicating the label. The dimensions for the input layer and the output layer are N_{in} and 10, respectively. First, we compute the first principle direction. Transform each image by

$$\vec{x}'_j = \vec{x}_j - \frac{1}{n} \sum_{k=0}^{n-1} \vec{x}_k, \quad j = 0, 1, \dots, n-1. \quad (15)$$

Denote all images by $X = [\vec{x}'_0, \vec{x}'_1, \dots, \vec{x}'_{n-1}] \in \mathbb{R}^{N_{in} \times n}$. The covariance matrix is $C_x = XX^T$. The eigenvector of the maximal eigenvalue of C_x can be obtained, denoted by $\vec{p}_1 \in \mathbb{R}^{N_{in}}$, i.e., the first principle direction. The projection of each image in the \vec{p}_1 direction is $x'_k \triangleq \vec{p}_1^T \vec{x}'_k$. We rescale x'_k to x_k such that $x_k \in [0, 1]$ by

$$x_k = \frac{(x'_k - \min_j x'_j)}{\max_l (x'_l - \min_j x'_j)}. \quad (16)$$

Then, the sample set is $S = \{(x_0, \vec{y}_0), (x_1, \vec{y}_1), \dots, (x_{n-1}, \vec{y}_{n-1})\}$. For illustration, we only consider the first component of \vec{y} , i.e., $\vec{y}^{(1)}$. Note that now $\{x_k\}_{k=0}^{n-1}$ is a non-uniform sampling. Using non-uniform FFT (NUFFT), we can obtain

$$F[\vec{y}^{(1)}][\gamma_k] = \sum_{j=0}^{n-1} \vec{y}_k^{(1)} \exp(-2\pi i x_j k). \quad (17)$$

Then, the sampling on the Fourier domain is

$$S_\gamma = \{(\gamma_0, F[\vec{y}^{(1)}](\gamma_0)), (\omega_1, F[\vec{y}^{(1)}](\gamma_1)), \dots, (\gamma_{n-1}, F[\vec{y}^{(1)}](\gamma_{n-1}))\}. \quad (18)$$

After each training step, we feed each \vec{x}_k into the DNN and obtain the DNN output $\Upsilon(\vec{x}_k)$:

$$S_T = \{(x_0, \Upsilon(\vec{x}_0)), (x_1, \Upsilon(\vec{x}_1)), \dots, (x_{n-1}, \Upsilon(\vec{x}_{n-1}))\}. \quad (19)$$

Using non-uniform FFT (NUFFT), similarly, we can obtain the sampling of the DNN's first dimension output on the Fourier domain

$$S_{\Upsilon, \gamma} = \{(\gamma_0, F[\Upsilon^{(1)}](\gamma_0)), (\omega_1, F[\Upsilon^{(1)}](\gamma_1)), \dots, (\gamma_{n-1}, F[\Upsilon^{(1)}](\gamma_{n-1}))\}, \quad (20)$$

which is shown in Fig.2a. We then examine the relative error of certain selected important frequency components (marked by black squares). As shown in the first column in Fig.2b, we can observe that the DNN tends to capture low-frequency components first.

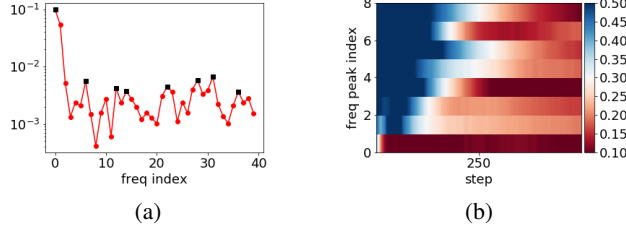


Figure 2: F-Principle with cross entropy loss on MNIST dataset. The Fourier coefficient amplitude of the first output dimension of the target function is shown in (a). Frequency peaks are marked by black dots. (b) Relative difference at different recording steps for different selected frequency indexes. The training data are 10000 test samples of MNIST dataset. We use a DNN with width 400-200 with batch size as 128, the output layer has 10 neurons with softmax as activation function, and learning rate is 10^{-5} . The parameters of the DNN are initialized following a Gaussian distribution with mean 0 and standard deviation 0.2.

4 Experiment: Poisson's equations

Consider one-dimension (1-d) Poisson's equation [11, 17]:

$$\begin{aligned} -\Delta u(x) &= g(x), \quad x \in \Omega = (-1, 1) \\ u(x) &= 0, \quad x = -1, 1. \end{aligned} \quad (21)$$

The Poisson's equation can be solved by numerical schemes (e.g., Jacobi method) or DNN. As well known, high frequency converges faster in the Jacobi method. In the following, we would show that high frequency converges slower when the DNN is applied to solve the above Poisson's equation.

4.1 Central differencing scheme and Jacobi method

$[-1, 1]$ is uniformly discretized into $n + 1$ points with step $\Delta x = 2/n$, i.e., x_0, x_1, \dots, x_n . The Poisson's equation in Eq. (21) can be solved by central differencing scheme:

$$-\Delta u_i = -\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} = g(x_i), \quad i = 1, 2, \dots, n. \quad (22)$$

Write the above in the matrix form:

$$Au = g, \quad (23)$$

where

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 \\ \vdots & \vdots & \dots & & & \vdots \\ 0 & 0 & \dots & 0 & -1 & 2 \end{pmatrix}_{(n-1) \times (n-1)}, \quad (24)$$

$$u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix}, \quad g = (\Delta x)^2 \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{n-2} \\ g_{n-1} \end{pmatrix}, \quad x_i = 2\frac{i}{n}. \quad (25)$$

If n is not a large number, Eq. (23) can be solved by performing the inverse of A . When n is a very large number, this problem can be solved by iterative schemes. For example, we illustrate the Jacobi method. Let $A = D - L - U$, where D is diagonal, and L and U are the strictly lower and upper parts of $-A$, respectively. Then, we can obtain

$$u = D^{-1}(L+U)u + D^{-1}g. \quad (26)$$

The Jacobi iteration is

$$u^{l+1} = D^{-1}(L+U)u^l + D^{-1}g. \quad (27)$$

We perform error analysis of the above iteration process. Denote u^* as the true value obtained by directly performing inverse of A in Eq. (23). The error at step $l+1$ is $e^{l+1} = u^{l+1} - u^*$. Then, $e^{l+1} = R_J e^l$, where $R_J = D^{-1}(L+U)$. The converging speed of e^l is determined by the eigenvalues of R_J , that is,

$$\lambda_k = \lambda_k(R_J) = \cos \frac{k\pi}{n}, \quad k = 1, 2, \dots, n-1, \quad (28)$$

and the corresponding eigenvector v_k is

$$v_{k,j} = \sin \frac{jk\pi}{n}, \quad j = 1, 2, \dots, n-1. \quad (29)$$

Write

$$e^l = \sum_{k=1}^{n-1} \alpha_k^l v_k, \quad (30)$$

where α_k^l can be understood as the magnitude of e^l in the direction of v_k . Then,

$$e^{l+1} = \sum_{k=1}^{n-1} \alpha_k^l R_J v_k = \sum_{k=1}^{n-1} \alpha_k^l \lambda_k v_k. \quad (31)$$

$$\alpha_k^{l+1} = \lambda_k \alpha_k^l.$$

Therefore, the converging speed of e^l in the direction of v_k is controlled by λ_k . Since

$$\cos \frac{k\pi}{n} = -\cos \frac{(n-k)\pi}{n}, \quad (32)$$

the frequencies k and $(n-k)$ are closely related and converge with the same speed. Consider the frequency $k < n/2$, λ_k is larger for lower frequency. Therefore, lower frequency converges slower in the Jacobi method.

4.2 DNN approach

Similar as the loss function in Ref [11], we consider the following loss function (energy method)

$$I(u) = \int_{\Omega} \left(\frac{1}{2} |\nabla_x u(x)|^2 - g(x)u(x) \right) dx + \beta \int_{\partial\Omega} u(x)^2 ds. \quad (33)$$

It is equivalent to solve Poisson's equation by finding the function that minimizes $I(u)$ (Dirichlet's principle) [17]. The last term in $I(u)$ is a penalty in order to satisfy the boundary condition. The DNN structure is 1-d input (i.e., x) and 1-d output (denoted as $\Upsilon(x)$) for solving Eq. (21). β is a constant.

The procedure is similar. We discretized $[-1, 1]$ into $n+1$ even-space points. In each training step, we compute $\Upsilon(x_i)$ for $i = 0, 1, 2, \dots, n$. The gradient of $I(\Upsilon)$ with respect to parameter Θ is

$$\frac{dI(\Upsilon)}{d\Theta} = \sum_{i=0}^n \frac{dI(\Upsilon(x_i))}{d\Theta}. \quad (34)$$

At each training step, we compare $\Upsilon(x)$ and $u^*(x)$ in the Fourier domain. Note that $u^*(x)$ is the one obtained by directly performing inverse of A in Eq. (23).

4.3 Experiment

Consider

$$g(x) = \sin(x) + 4 \sin(4x) - 8 \sin(8x) + 16 \sin(24x). \quad (35)$$

As shown in Fig. 3a, after training, the DNN output can well fit the solution u^* obtained by directly performing inverse of A in Eq. (23). As shown in Fig. 3b, there are three peaks of u^* in the Fourier domain.

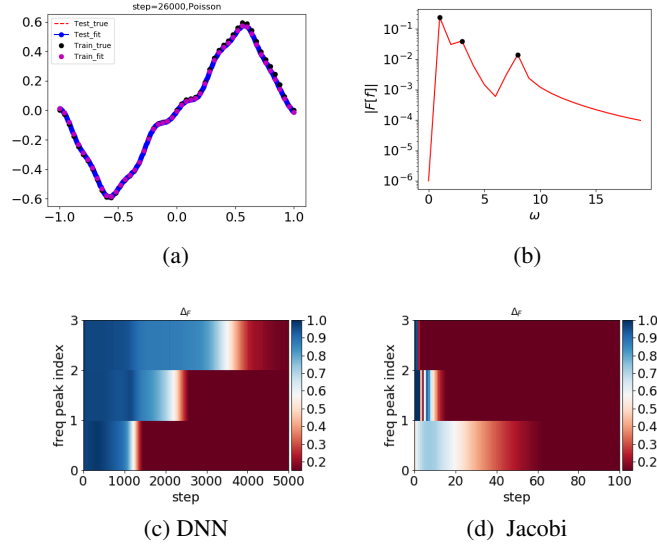


Figure 3: Frequency domain analysis of the Poisson's equation in Eq. (21) with $g(x) = \sin(x) + 4\sin(4x) - 8\sin(8x) + 16\sin(24x)$. (a) $u(x)$. The true value is computed by central differencing scheme with directly compute the inverse of coefficient matrix. (b) $|F[u^*]|$ (red solid line) as a function of frequency index. Frequency peaks are marked by black dots. (c, d) Relative difference at different recording steps for different selected frequency indexes. The training data and the test data are evenly sampled in $[-1, 1]$ with sample size 51 and 401, respectively. We use a DNN with width 4000-800 with full batch training. The learning rate is 5×10^{-6} at beginning and halved every 10000 training epochs. β is 10. Each step consists of four epochs. (d) is the result of Jacobi iteration. The parameters of the DNN are initialized following a Gaussian distribution with mean 0 and standard deviation 0.05.

To examine the convergence behavior of different frequency components during the DNN training, we compute the relative difference of the DNN output $\Upsilon(x)$ and $u^*(x)$ in frequency domain at each recording step, i.e.,

$$\Delta_F(\gamma) = \frac{|F[u^*](\gamma) - F[\Upsilon](\gamma)|}{|F[u^*](\gamma)|}. \quad (36)$$

As shown in Fig. 3c, F-Principle holds well in solving Poisson's equation [3, 4]. For comparison, we also show that low frequency converges much slower than high frequency in Jacobi method, as shown in Fig. 3d.

5 Combination of DNN and convention methods

In light of the above numerical simulations, it is natural to consider if we can combine DNN and Jacobi method to solve the Poisson's equation. For simplicity, we call the combination method D-Jacobi method.

In the first part of D-Jacobi method, we solve the Poisson's equation by DNN with M steps. In the second part, we use the DNN output at step M as the initial value for the Jacobi method.

We solve the problem in Fig. 3 by a laptop (Dell, Precision 5510). As shown in Fig.4a, the DNN loss fluctuates after some running time. We use Jacobi method to solve the problem after some time points, which are indicated by vertical dashed lines. As shown in Fig.4b (Fig.4c), green stars indicates the $|\Upsilon - u^*|_\infty$ of the DNN output at different steps. Dashed lines indicates the evolution of the Jacobi (Gauss-Seidel) method. As we can see, if the selected timing is too early, it would still take long time to converge to a small error, because the low frequencies are not converged, yet. If the selected timing is too late, much time would be waste because the DNN is hard to capture high frequencies

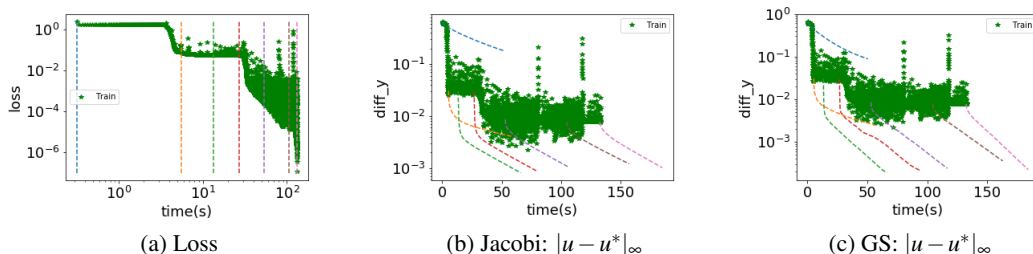


Figure 4: Combination methods for solving the Poisson’s equation in Eq. (21) with $g(x) = \sin(x) + 4\sin(4x) - 8\sin(8x) + 16\sin(24x)$. The abscissa is the computer running time. (a) Loss is the form in Eq. (33). We use Jacobi method to solve the problem after several time points, which are indicated by vertical dashed lines. (b) Green stars indicates the $|Y - u^*|_\infty$ at different steps. Dashed lines indicates the evolution of the Jacobi method. (c) Gauss-Seidel method. The training data are evenly sampled in $[-1, 1]$ with sample size 1001. u^* is computed by central differencing scheme with directly computing the inverse of coefficient matrix. We use a DNN with width 4000-500-400 with full batch training, and learning rate is 5×10^{-4} . β is 10. The parameters of the DNN are initialized following a Gaussian distribution with mean 0 and standard deviation 0.02.

and fluctuates a lot. The selected timing of the green or the red one is a better choice. In practice, a better way to select the timing is when the loss gets flat and fluctuated for a short while.

6 Discussion

In this work, we have shown that F-Principle holds well in the DNN training with a general loss function, extending the study of F-Principle in the loss function of mean square error in previous works [3, 4, 5]. Along with the previous study that F-Principle holds in both DNN and convolutional neural networks with the activation function of either tanh or Relu [3, 4], these works implicate that the F-Principle may provide understandings to the generalization ability of general DNNs.

We also show that the generality of F-Principle in the DNN training could potentially be useful in designing algorithms for solving practical problems. To be specific, we apply DNN to solve 1-d Poisson’s equation. Compared with conventional numerical schemes, DNN could potentially work better in rather high dimensions [11]. In addition, it does not requires discretization for the DNN method, which would be much easier to be implemented. In future, it would be interested to use DNN’s F-Principle to develop numerical schemes for solving various problems which would benefit from a fast converging of low frequency.

Acknowledgments

The author wants to thank Weinan E, Wei Cai for helpful discussion. The author also wants to thank Tao Luo, Zheng Ma, Yanyang Xiao and Yaoyu Zhang for the discussion of the F-Principle. This work was funded by the NYU Abu Dhabi Institute G1301.

References

- [1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
- [2] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

- [3] Zhi-Qin J Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. *arXiv preprint arXiv:1807.01251*, 2018.
- [4] Zhiqin John Xu. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.
- [5] Nasim Rahaman, Devansh Arpit, Aristide Baratin, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of deep neural networks. *arXiv preprint arXiv:1806.08734*, 2018.
- [6] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.
- [7] T Poggio, K Kawaguchi, Q Liao, B Miranda, L Rosasco, X Boix, J Hidary, and HN Mhaskar. Theory of deep learning iii: the non-overfitting puzzle. Technical report, CBMM memo 073, 2018.
- [8] Guillermo Valle Pérez, Ard A Louis, and Chico Q Camargo. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.
- [9] Daniel Jakubovitz, Raja Giryes, and Miguel RD Rodrigues. Generalization error in deep learning. *arXiv preprint arXiv:1808.01174*, 2018.
- [10] Lei Wu, Zhanxing Zhu, and Weinan E. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.
- [11] E Weinan and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.
- [12] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *arXiv preprint arXiv:1707.03351*, 2017.
- [13] E Weinan, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.
- [14] William L Briggs, Steve F McCormick, et al. *A multigrid tutorial*, volume 72. Siam, 2000.
- [15] Gang Bao, Peijun Li, Junshan Lin, and Faouzi Triki. Inverse scattering problems with multi-frequencies. *Inverse Problems*, 31(9):093001, 2015.
- [16] Alex Barnett, Leslie Greengard, Andras Pataki, and Marina Spivak. Rapid solution of the cryo-em reconstruction problem by frequency marching. *SIAM Journal on Imaging Sciences*, 10(3):1170–1195, 2017.
- [17] Lawrence C Evans. *Partial differential equations*. 2010.