Understanding training and generalization in deep learning by Fourier analysis[*]

Zhi-Qin John Xu[†]

Abstract. **Background**: It is still an open research area to theoretically understand why Deep Neural Networks (DNNs)—equipped with many more parameters than training data and trained by (stochastic) gradient-based methods—often achieve remarkably low generalization error. **Contribution**: We study DNN training by Fourier analysis. Our theoretical framework explains: i) DNN with (stochastic) gradient-based methods endows low-frequency components of the target function with a higher priority during the training; ii) Small initialization leads to good generalization ability of DNN while preserving the DNN's ability of fitting any function. These results are further confirmed by experiments of DNNs fitting the following datasets, i.e., natural images, one-dimensional functions and MNIST dataset.

傅里叶分析DNN在训练中的情况。1.首先拟合低频率的部分，高频率的部分之后拟合
2.小的初始化效果更好。 我觉得这是废话......

Key words. deep learning, generalization, training, frequency, optimization

AMS subject classifications. 68Q32, 68T01,

1. Introduction.

Background. Deep learning has achieved great success as in many fields [12]. Recent studies have focused on understanding why DNNs, trained by (stochastic) gradient-based methods, can generalize well, that is, DNNs often fit the test data well which are not used for training in practice. Counter-intuitively, although DNNs have many more parameters than training data, they can rarely overfit the training data in practice. 违反常理地，DNN那么多参数，但是过拟合问题却不严重

Several studies have focused on the local property (sharpness/flatness) of loss function at minima [7] to explore the DNN's generalization ability. Keskar et al., (2016) [9] empirically demonstrated that with small batch in each training step, DNNs consistently converge to flat minima, and lead to a better generalization. However, Dinh et al., (2017) [4] argued that most notions of flatness are problematic. To this end, Dinh et al., (2017) [4] used deep networks with rectifier linear units (ReLUs) to theoretically show that any minimum can be arbitrarily sharp or flat without specifying parameterization. With the constraint of small weights in parameterization, Wu et al., (2017) [19] proved that for two-layer ReLU networks, low-complexity solutions lie in the areas with small Hessian, that is, flat and large basins of attractor [19]. They then concluded that a random initialization tends to produce starting parameters located in the basin of flat minima with a high probability, using gradient-based methods.

Several studies rely on the concept of stochastic approximation or uniform stability [2, 6]. To ensure the stability of a training algorithm, Hardt et al., (2015) [6] assumed loss function with good properties, such as Lipschitz or smooth conditions. However, the loss function of a DNN is often very complicated [22].

Another approach to understand the DNN's generalization ability is to find general principles during training. Empirically, Arpit et al., (2017) [1] suggested that DNNs may learn simple patterns first in real data, before memorizing. Xu et al., (2018) [20] empirically found a similar phenomenon,

[*]
Funding: This work was funded by the NYU Abu Dhabi Institute G1301

[†]New York University Abu Dhabi, United Arab Emirates and Courant Institute of Mathematical Sciences, New York University, New York, United States. (zhiqinxu@nyu.edu)

频率原则：对于一个低频成分占主要部分的函数，DNN优先拟合低频成分，先减少DNN和高频成分之间的距离，之后再慢慢地拟合高频的成分，

which is referred to *Frequency Principle (F-Principle)*, that is, for a low-frequency dominant function, DNNs with common settings first quickly capture the dominant low-frequency components while keeping the amplitudes of high-frequency components small, and then relatively slowly captures those high-frequency components. F-Principle can explain how the training can lead DNNs to a good generalization empirically [20]. However, without a theoretical understanding, Xu et al., (2018) [20] are limited to a understanding of DNNs' fitting of low-frequency dominate functions with low dimension empirically[1].

Contribution. In this work, we develop a theoretical framework in Fourier domain aiming to understanding the training process and the generalization of DNNs with sufficient neurons and hidden layers. We show that for any parameter, the gradient descent magnitude in each frequency component of the loss function is proportional to the product of two factors: one is a decay term with respect to (w.r.t.) frequency; the other is the amplitude of the difference between the DNN output and the target function. This theoretical framework can answer the following crucial questions:

DNN关于频率的梯度 正比于 距离目标的误差大小 X 有关于频率的衰减项，频率越大，衰减的越严重

Question 1: Do DNNs trained by gradient-based methods endow low-frequency components with higher priority during the training process? The *decay term* in the gradient descent magnitude (see Eq. (3.13)) shows that the priority of low-frequency components dominates high-frequency components before converging. The theoretical understanding of this problem can explain the F-Principle observed in the previous study [20]. In addition, this theoretical framework can also explain the more complicated behavior of DNNs' fitting of high-frequency dominant functions.

Question 2: How does initialization affect the DNN generalization? We analyze the DNN with the sigmoid activation function. We begin from the following point: the power spectrum of the sigmoid function exponentially decays w.r.t. frequency, in which the exponential decay rate is proportional to the inverse of weight. We then show that small (large) initialization would result in small (large) amplitude of high-frequency components, thus leading the DNN output to a low (high) complexity function with good (bad) generalization ability. Therefore, with small initialization, sufficient large DNNs can fit any function [3] while keeping good generalization. 小的初始化最后可以有更好的效果，因为振幅更小，泛华性能好

We demonstrate that the analysis in this work can be qualitatively extended to general DNNs. We exemplified our theoretical results through DNNs fitting natural images, 1-d functions and MNIST dataset [11].

The paper is established as follows. The common settings of DNNs in this work are presented in Section 2. The theoretical framework is given in Section 3. We show that the mean magnitude of DNN parameters only change slightly during the DNN training empirically in Section 4. The theoretical framework is applied to understand the training and generalization of DNNs in Section 5. The conclusions and discussions are followed in Section 6.

2. Methods.

The activation function for each neuron is tanh. We use DNNs of multiple hidden layers with no activation function for the output layer. The DNN is trained by Adam optimizer [10]. Parameters of the Adam optimizer are set to the default values [10]. The loss function is the mean squared error of the difference between the DNN output and the target function in the training set.

[1]After our submission, we found there is another paper [16] found a similar result that Lower Frequencies are Learned First and gave some theoretical understanding. However, the theory in Rahaman et al., [16] is incomplete. See more in our *Discussion* "Weight norm".

3. Theoretical framework. In this section, we will develop a theoretical framework in the Fourier domain to understand the training process of DNN. For illustration, we first use a DNN with one hidden layer with sigmoid function $\sigma(x)$ as the activation function:

$$\sigma(x) = \frac{e^x}{e^x+1}, \quad x \in \mathbb{R}.$$

The Fourier transform of $\sigma(ax+b)$ with $a, b \in \mathbb{R}$ is (See Appendix A),

$$F[\sigma(ax+b)](\omega) = -\frac{i\pi}{|a|}\exp(ib\omega/a)\frac{2}{\exp(\pi\omega/a)-\exp(-\pi\omega/a)}, \tag{3.1}$$

where

$$F[\sigma(x)](\omega) = \int_{-\infty}^{\infty}\sigma(x)\exp(-i\omega x)\mathrm{d}x.$$

When $\omega = 0$, $F[\sigma(ax+b)](\omega)$—the integration of $\sigma(x)$ on $[-\infty, \infty]$—would be infinity. However, we only consider a finite range of x in practice. Thus, the infinity at $\omega = 0$ in Eq. (3.1) is not an issue in practice.

Consider a DNN with one hidden layer with N nodes, 1-d input x that has a 1-d output:

$$T(x) = \sum_{j=1}^{N}\alpha_j\sigma(a_jx+b_j), \quad \alpha_j, a_j, b_j \in \mathbb{R}. \tag{3.2}$$

1-N-SIGMOID-1的神经网络

Note that we call all a_j, α_j and b_j as *parameters*, in which a_j and α_j are *weights*, and b_j is a *bias term*. When $|\pi\omega/a_j|$ is large, without loss of generality, we assume $\pi\omega/a_j \gg 0$ and $a_j > 0$,

$$F[T](\omega) \approx \sum_{j=1}^{N}\alpha_j\left[-\frac{2i\pi}{a_j}\exp(ib_j\omega/a_j)\exp(-\pi\omega/a_j)\right]. \tag{3.3}$$

对网络做傅里叶变换

We define the difference between DNN output and any *target function* $f(x)$ at each ω as

$$D(\omega) \triangleq F[f](\omega) - F[T](\omega).$$

误差项

Write $D(\omega)$ as

$$D(\omega) = A(\omega)e^{i\theta}, \tag{3.4}$$

误差项写成极坐标形式

where $A(\omega)$ and θ indicate the amplitude and phase of $D(\omega)$, respectively.

The loss at frequency ω is $L(\omega) = |D(\omega)|^2$, where $|\cdot|$ denotes the norm of a complex number. The total loss function is defined as:

$$L = \sum_{\omega}L(\omega).$$

loss=|D|^2

Note that according to the Parseval's theorem[2], this loss function in the Fourier domain is equal to the commonly used loss of mean squared error, that is,

$$L = \frac{1}{N}\sum_{x}(f(x)-T(x))^2, \tag{3.5}$$

[2]Without loss of generality, the constant factor that is related to the definition of corresponding Fourier Transform is ignored here.

At frequency ω, the amplitude of the gradient with respect to each parameter can be obtained,

$$\left|\frac{\partial L(\omega)}{\partial \alpha_j}\right| = \left|\frac{2\pi}{a_j}\right| J(\omega, a_j)\,|1-C_1|, \tag{3.6}$$

$$\left|\frac{\partial L(\omega)}{\partial a_j}\right| = \left|\frac{2\pi\alpha_j}{a_j^2}\right| J(\omega, a_j) C_2, \tag{3.7}$$

$$\left|\frac{\partial L(\omega)}{\partial b_j}\right| = \left|\frac{2\pi\alpha_j}{a_j^2}\omega\right| J(\omega, a_j)\,|1-C_1|, \tag{3.8}$$

where

$$J(\omega, a_j) = A(\omega)\exp\left(-|\pi\omega/a_j|\right), \tag{3.9}$$

$$C_1 = \exp\left(2i\left(b_j\omega/a_j - \theta\right)\right), \tag{3.10}$$

$$C_2 = \left|\left[(ib_j+\pi)\omega/a_j - 1\right] - C_1\left[(-ib_j+\pi)\omega/a_j - 1\right]\right|. \tag{3.11}$$

The descent amount at any direction, say, with respect to parameter Θ_{jk}, is

$$\frac{\partial L}{\partial \Theta_{jk}} = \sum_{\omega} \frac{\partial L(\omega)}{\partial \Theta_{jk}}. \tag{3.12}$$

The absolute contribution from frequency ω to this total amount at Θ_{jk} is 对于各个参数的通用形式

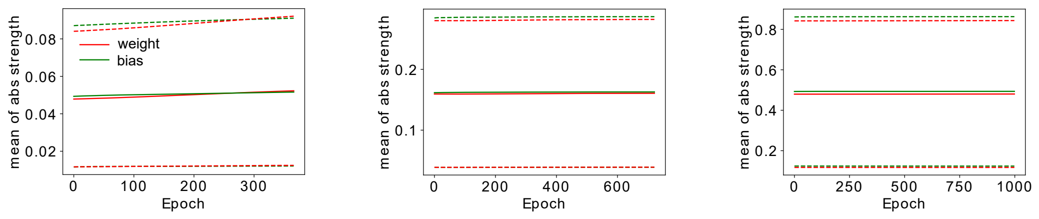
$$\left|\frac{\partial L(\omega)}{\partial \Theta_{jk}}\right| = A(\omega)\exp\left(-|\pi\omega/a_j|\right) G_{jk}(\Theta_j, \omega), \tag{3.13}$$

这个公式解释了：loss的梯度下降法与频率的关系明显：频率越小w小，梯度越大，所以优先拟合低频成分

where $\Theta_j \triangleq \{a_j, b_j, \alpha_j\}$, $\Theta_{jk} \in \Theta_j$, $G_{jk}(\Theta_j, \omega)$ is a function with respect to Θ_j and ω, which can be found in one of Eqs. (3.6, 3.7, 3.8).

When the component at frequency ω does not converge yet, $\exp\left(-|\pi\omega/a_j|\right)$ would dominate $G_{jk}(\Theta_j, \omega)$ for a small a_j. After training, when the component at frequency ω converges, $A(\omega) \approx 0$. The contribution of frequency ω to the total descent amount vanishes. Therefore, the behavior of Eq. (3.13) is dominated by $A(\omega)\exp\left(-|\pi\omega/a_j|\right)$. This dominant term also indicates that weights are much more important than bias terms, which will be verified by MNIST dataset later.

Next, we demonstrate that the analysis of $A(\omega)\exp\left(-|\pi\omega/a_j|\right)$ can be qualitatively extended to general DNNs. $A(\omega)$ in Eq. (3.13) comes from the square operation in the loss function, thus, is irrelevant with DNN structure. $\exp\left(-|\pi\omega/a_j|\right)$ in Eq. (3.13) comes from the exponential decay of the activation function in the Fourier domain. The analysis of $\exp\left(-|\pi\omega/a_j|\right)$ is insensitive to the following factors. i) Activation function. The power spectrum of most activation functions decreases as the frequency increases; ii) Neuron number. The summation in Eq. (3.2) does not affect the exponential decay; iii) Multiple hidden layers. If there are multiple hidden layers, the composition of continuous activation functions is still a continuous function. The power spectrum of the continuous function still decays as the frequency increases; iv) High-dimensional input. We simply need to consider the vector form of ω in Eq. (3.1); v) High-dimensional output. The total loss function is the summation of the loss of each output node. Therefore, the analysis of $A(\omega)\exp\left(-|\pi\omega/a_j|\right)$ of a single hidden layer qualitatively applies to different activation functions, neuron numbers, multiple hidden layers, and high-dimensional functions.



(a) std: 0.06 (b) std: 0.2 (c) std: 0.6

Figure 1: Magnitude of DNN parameters during fitting MNIST dataset. DNN parameters are initialized by Gaussian distribution with mean 0 and standard deviation 0.06, 0.2, 0.6 for (a, b, c), respectively. Solids lines show the mean magnitude of the absolute weights (red) and the absolute bias (green) at each training epoch. The dashed lines are the mean$\pm$std for the corresponding color. Note that the green and the red lines almost overlap. We use a tanh DNN with width: 800-400-200-100. The learning rate is 10^{-5} with batch size 400.

4. The magnitude of DNN parameters during training.

Since the magnitude of DNN parameters is important to the analysis of the gradients, such as a_j in Eq. (3.13), we study the evolution of the magnitude of DNN parameters during training. Through training DNNs by MNIST dataset, empirically, we show that for a network with sufficient neurons and layers, the mean magnitude of the absolute values of DNN parameters only changes slightly during the training. For example, we train a DNN by MNIST dataset with different initialization. In Fig.1, DNN parameters are initialized by Gaussian distribution with mean 0 and standard deviation 0.06, 0.2, 0.6 for (a, b, c), respectively. We compute the mean magnitude of absolute weights and bias terms. As shown in Fig.1, the mean magnitude of the absolute value of DNN parameters only changes slightly during the training. Thus, empirically, the initialization almost determines the magnitude of DNN parameters. Note that the magnitude of DNN parameters can have significant change during training when the network size is small (We have more discussion in *Discussion*).

5. Understanding deep learning.

Here, we consider only DNNs with sufficient neurons and layers. Since initialization is very important to deep learning, we will first discuss small initialization and then discuss different situations of initialization.

5.1. Fitting low-frequency dominant functions.

First, we use Eq. (3.13) to understand the F-Principle observed in Xu et al., (2018) [20] (See *Introduction*). To show that the F-Principle holds in real data, we train a DNN to fit a natural image, as shown in Fig.2a—a mapping from position (x,y) to gray scale strength, which is subtracted by its mean and then normalized by the maximal absolute value. As an illustration of F-Principle, we study the Fourier transform of the image with respect to x for a fixed y (red dashed line in Fig.2a, denoted as the *target function* $f(x)$ in spatial domain). In a finite interval, the frequency components of the target function can be quantified by Fourier coefficients computed from Discrete Fourier Transform (DFT). Note that the frequency in DFT is discrete. The Fourier coefficient of $f(x)$ for the frequency component ω is denoted by $F[f](\omega)$ (a complex number in general). $|F[f](\omega)|$ is the corresponding amplitude. Fig.2b displays the first

40 frequency components of $|F[f](\omega)|$. To examine the convergence behavior of different frequency components during the training, we compute the relative difference of the DNN output and $f(x)$ in frequency domain at *each recording step*, that is,

$$\Delta_F(\omega) = \frac{|F[f](\omega) - F[T](\omega)|}{|F[f](\omega)|}, \tag{5.1}$$

where T denotes the DNN output. As shown in Fig.2c, the first four frequency peaks converge from low to high in order.

The mechanism underlying the phenomenon above is as follows. With small initialization, at the beginning of the training, the DNN output T is close to zero and $|F[T](\omega)|$ is also close to zero. Therefore, $A(\omega)$ in Eq. (3.4)—the amplitude of difference between the DNN output and the target function—is close to $|F[f](\omega)|$, that is, the gradient descent is proportional to $|F[f](\omega)|$. As the definition in Eq. (5.1), the relative difference $\Delta_F(\omega)$ is then independent of $|F[f](\omega)|$ at the beginning. Due to the small initialization, $\exp(-|\pi\omega/a_j|)$ in Eq. (3.13) has a large decay rate $|\pi/a_j|$. Therefore, the descent direction is mainly determined by low-frequency components. Initially, the convergence speed for $\Delta_F(\omega)$ exponentially decays as the frequency increases. When the DNN is fitting low-frequency components, high-frequency components stay small compared with low-frequency components due to the following reason—DNN parameters are small throughout the training, thus, $\exp(-|\pi\omega/a_j|)$ in Eq. (3.3) leads to that high frequency components of each neuron are small during low-frequency convergence; The gradient descent does not drive the phase of high-frequency components, i.e., $\exp(ib_j\omega/a_j)$ in Eq. (3.3), towards any specific direction; Therefore, until low-frequency converged, the amplitude of high-frequency components of the DNN output is a summation of small values with random phase. As an example in Fig.2d, when the DNN is fitting low-frequency components, high-frequency components stay relatively small.

When a low-frequency component converges, say, ω_L, $A(\omega_L) \approx 0$ for the low-frequency components, the main contributor to the total descent amount will be higher frequency components, say, ω_H. The contribution from frequency ω_H to the descent amount is constrained by $A(\omega_H)\exp(-|\pi\omega_H/a_j|)$. The gradient descent of ω_H would cause DNN output to deviate from the target function at ω_L. If the DNN differs too much at ω_L again, the $\exp(-|\pi\omega_L/a_j|)$ would dominate the total gradient descent, leading to the convergence of ω_L. The maximum deviation of the frequency component at ω_L occurs approximately when the decent amount caused by ω_H and ω_L are comparable, that is,

$$A(\omega_L)\exp(-|\pi\omega_L/a_j|) \approx A(\omega_H)\exp(-|\pi\omega_H/a_j|), \tag{5.2}$$

then, we have

$$A(\omega_L) \approx A(\omega_H)\exp(-|\pi(\omega_H - \omega_L)/a_j|). \tag{5.3}$$

Since $A(\omega_H) \ll |F[f](\omega_L)|$ for low-frequency dominant functions, a_j is close to 0 and $\omega_H > \omega_L$, then, the deviation at ω_L, $A(\omega_L)$, is much smaller than $|F[f](\omega_L)|$. Therefore, the descent amount caused by ω_H only causes a small fluctuation around $|F[f](\omega_L)|$. That is, when the DNN is fitting higher frequency components, the low-frequency components stay converged as an example shown in Fig.2c.

Viewing from the snapshots during the training process, we can see DNN captures the image from coarse-grained low-frequency components to detailed high-frequency components (Fig.2e-2g). And the DNN output generalizes well (Fig.2h), that is, the test error is very close to the training error [8].

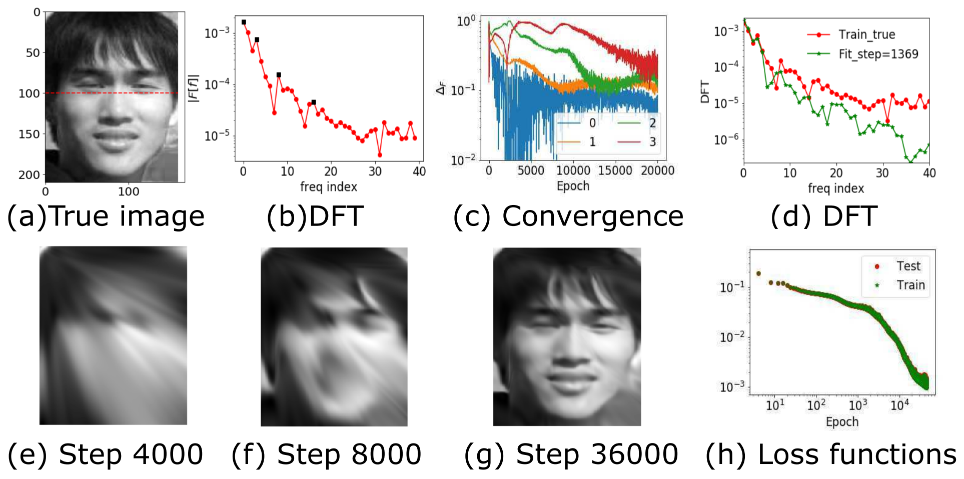


Figure 2: Convergence from low to high frequency for a natural image. The training data are all pixels whose horizontal indexes are odd. (a) True image. (b) $|F[f]|$ of the red dashed pixels in (a) as a function of frequency index—Note that for DFT, we can refer to a frequency component by the *frequency index* instead of its physical frequency.—with selected peaks marked by black dots. (c) Δ_F at different training epochs for different selected frequency peaks in (b). (d) $|F[f]|$ (red) and $|F[T]|$ (green) at epoch 1369. (e-g) DNN outputs of all pixels at different training steps. (h) Loss functions. We use a DNN with width 500-400-300-200-200-100-100. We train the DNN with the full batch and learning rate 2×10^{-5}. We initialize DNN parameters by Gaussian distribution with mean 0 and standard deviation 0.08.

The analysis for the image in Fig.2 is the similar for the famous image "Lena", as shown in Fig.5 in Appendix. Note that in Fig.2e and 2f, left-upper corner is better fitted than right-lower corner. This is due to the small initialization of bias terms. By use a larger initialization of bias terms in image "Lena", as shown in Fig.5 in Appendix, this bias fitting disappears. In addition, this theoretical framework can also explain the more complicated training behavior of DNNs' fitting of high-frequency dominant functions (See Appendix B).

5.2. Initialization and generalization. Next, we analyze how initialization can affect the DNN's generalization ability. We first perform schematic analysis on why small initialization of DNN's parameters can lead to a good generalization while a large initialization leads to a poor generalization. We then examine our analysis by training DNNs to fit MNIST dataset and a natural image.

5.2.1. Schematic analysis. Different initialization can result in very different generalization ability of DNN's fitting and very different training courses. Here, we schematically analyze DNN's output after training. With finite training data points, there is an effective frequency range for this training set, which is defined as the range in frequency domain bounded by Nyquist-Shannon sampling theorem [17] when the sampling is evenly spaced, or its extensions [21, 13] otherwise. Based on the effective frequency range, we can decompose the Fourier transform of DNN's output into two parts, that is, effective frequency range and *extra-higher* frequency range. For different initialization, since the DNN can well fit the training data, the frequency components in the effective frequency range are the same.

Then, we consider the frequency components in the extra-higher frequency range. The amplitude at each frequency for node j is controlled by $\exp(-|\pi\omega/a_j|)$ with a decay rate $|\pi/a_j|$. This decay rate is large for small initialization. Since the gradient descent does not drive these extra-higher frequency components towards any specific direction, the amplitudes of the DNN output in the extra-higher frequency range stay small. For large initialization, the decay rate $|\pi/a_j|$ is relative small. Then, extra-higher frequency components of the DNN output could have large amplitudes and much fluctuate compared with small initialization.

Higher-frequency function is of more complexity (for example, using $\mathbb{E}\|\nabla_x f\|_2^2$ to characterize complexity [19]). With small (large) initialization, the DNN's output is a low-complexity (high-complexity) function after training. When the training data captures all important frequency components of the target function, a low-complexity DNN output can generalize much better than a high-complexity DNN output.

5.2.2. Experiments: MNIST dataset. We next use MNIST dataset to verify the effect of initialization. We use Gaussian distribution with mean 0 to initialize DNN parameters. For simplicity, we use *two-dimensional vector* $(\cdot,\cdot)$ to denote standard deviations of weights and bias terms, respectively. Fix the standard deviation for bias terms, we consider the effect of different standard deviations of weights, that is, $(0.01, 0.01)$ in Fig.3a and $(0.3, 0.01)$ in Fig.3b. As shown in Fig.3, in both cases, DNNs have high accuracy for the training data. However, compared the red dashed line in Fig.3a with the yellow dashed line in Fig.3b, for the small standard deviation of weight, the prediction accuracy of the test data is much higher than that of the large one.

Note that the effect of initialization is governed by weights rather than bias terms. To verify this, we initialize bias terms with standard deviation 2.5. As shown by black curves in Fig.3a, the DNN with standard deviation $(0.01, 2.5)$ has a bit slower training course and a bit lower prediction accuracy, which is suggested by Eq. (3.10) and Eq. (3.11).

5.2.3. Experiments: natural image. When the DNN is fitting the natural image in Fig.2a with small initialization, it generalizes well as shown in Fig.2h. Here, we show the case of large initialization. Except for a larger standard deviation, other parameters are the same as in Fig.2. After training, the DNN can well capture the training data, as shown in the left in Fig.4a. However, the DNN output at the test pixels are very noisy, as shown in the right in Fig.4a. The loss functions of the training data and the test data in Fig.4b show that the DNN generalizes poorly. To visualize the high-frequency components of DNN output after training, we study the pixels at the red dashed lines in Fig.4a. As shown in the left in Fig.4c, the DNN accurately fits the training data. However, for the test data, the DNN output fluctuates a lot, as shown in the right in Fig.4c. The poor generalization and the highly fluctuated DNN output are consistent with our theoretical analysis.

6. Discussions.

In this work, we have theoretically analyzed the DNN training process with (stochastic) gradient-based methods through Fourier analysis. Our theoretical framework explains the training process when the DNN is fitting low-frequency dominant functions or high-frequency dominant functions (See Appendix B). Based on the understanding of the training process, we explains

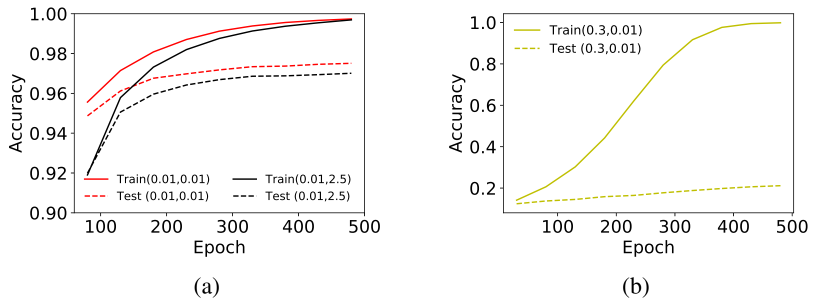


Figure 3: Analysis of the training process of DNNs with different initialization while fitting MNIST dataset. Illustrations are the prediction accuracy on the training data and the test data at different training epochs. We use a tanh DNN with width: 800-400-200-100. The learning rate is 10^{-5} with batch size 400. DNN parameters are initialized by Gaussian distribution with mean 0. The legend $(\cdot,\cdot)$ denotes standard deviations of weights and bias terms, respectively.

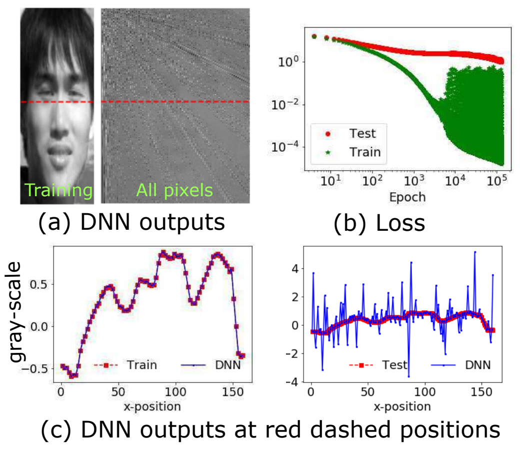


Figure 4: Analysis of the training process of DNNs with large initialization while fitting the image in Fig.2a. The weights of DNNs are initialized by a Gaussian distribution with mean 0 and standard deviation $(0.5, 0.08)$. (a) The DNN outputs at the training pixels (left) and all pixels (right). (b) Loss functions. (c) DNN outputs at the red dashed position in (a). We use a DNN with width 500-400-300-200-200-100-100, and learning rate 2×10^{-5}.

why DNN with small initialization can achieve a good generalization ability. Our theoretical result shows that the initialization of weights rather than bias terms mainly determines the DNN training and generalization. We exemplify our results through natural images and MNIST dataset. These analyses are not constrained to low-dimensional functions. Next, we will discuss the relation of our results with other studies and some limitations of these analyses.

Weight norm. In this work, we analyze the DNN with sufficient neurons and layers such that the mean magnitude of DNN parameters keeps almost constant throughout the training. However, if we use a small-scale network, the mean magnitude of DNN parameters often increases to a stable value. Empirically, we found that the training increases the magnitude of DNN parameters when the DNN is fitting high-frequency components, for which our theory can provide some insight. If the weights are too small, the decay rate of $\exp(-|\pi\omega/a_j|)$ in Eq. (3.3) will be too large. The training will have to increase the weights fit the high-frequency components of the target function because the number of neurons are fixed. By imposing regularization on the norm of weights in a small-scale network, we can prevent the training from fitting high-frequency components. Since high-frequency components usually have small power and are easily affected by noise, without fitting high-frequency components, the norm regularization will improve the DNN generalization ability. This discussion is consistent with other studies [15, 14]. We will address this topic about the evolution of the mean magnitude of DNN parameters of small-scale networks in our the future work.

Another work [16] studied the F-Principle (or called spectral bias in Rahaman et al., (2018) [16]) by using the fact that the overall trend for the spectral norm[3] increases gradually during training with a small-size DNN. Rahaman et al., (2018) estimated an inequality that the amplitude of each frequency component of the DNN output is controlled by the spectral norm of DNN weights. Based on this inequality, this implies that "longer training allows the network to represent more complex functions by allowing it to also fit higher frequencies". However, for a large-size DNN, the spectral norm almost does not change during the training (See an example in Fig.7 in Appendix.), that is, the bound of the amplitude of each frequency component of the DNN output almost does not change during the training. Then, the inequality in Rahaman et al., (2018) can not explain why the F-Principle still holds for a large-size DNN.

Loss function and activation function. In this work, we use the mean square error as loss function and tanh as activation function for the training. Empirically, we found that by using the mean absolute error as loss function, we can still observe the convergence order from low to high frequency when the DNN is fitting low-frequency dominant functions. The term $A(\omega)$ can be replaced by other forms that can characterize the difference between DNN outputs and the target function, by which the analysis of $A(\omega)$ can be extended. The key exponential term $\exp(-|\pi\omega/a_j|)$ comes from the property of the activation function. Note that when computing the Fourier transform of the activation function, we have ignored the windowing's effect – which would not change the property of the activation function in the Fourier domain whose power decays as the frequency increases. Therefore, for any activation function where power decreases as the frequency increases and any loss function which characterizes the difference of DNN outputs and the target function, the analysis in this work can be qualitatively extended. The exact mathematical forms of different activation functions and loss functions can be different. We leave this analysis to our future work.

Sharp/flat minima and generalization. Since $\exp(-|\pi\omega/a_j|)$ exists in all gradient forms in Eqs. (3.6, 3.7, 3.8), $\exp(-|\pi\omega/a_j|)$ will also exist in the second-order derivative of the loss function with respect to any parameter. Here, we only consider DNN parameters with similar magnitude. With smaller weights at a minima, the DNN has a good generalization ability along with that the second-order derivative at the minima is smaller, that is, a flatter minima. When the weights are very large, the minima is very sharp. When the DNN is close to a very sharp minima, one training step can cause the loss function deviate from the minima significantly (See an conceptual sketch in Figure 1

[3]In Rahaman et al., (2018) [16], "for matrix-valued weights, their spectral norm was computed by evaluating the eigenvalue of the eigenvector obtained with 10 power iterations. For vector-valued weights, we simply use the L_2 norm".

of [9]). We also observe that in Fig.4, for large initialization, the loss fluctuates significantly when it is small. Our theoretical analysis qualitatively shows that a flatter minima is associated with a better DNN generalization, which resembles the results of other studies [9, 19] (see *Introduction*).

Early stopping . Our theoretical framework through Fourier analysis can well explain F-Principle, in which DNN gives low-frequency components with higher priority as observed in Xu et al., (2018) [20]. Thus, our theoretical framework can provide insight into early stopping. High-frequency components often have low power [5] and are noisy, but with early stopping, we can prevent DNN from fitting high-frequency components to achieve a better generalization.

Noise and real data. Empirical studies found the qualitative differences in gradient-based optimization of DNNs on pure noise vs. real data [22, 1, 20]. We would discuss the mechanism underlying these qualitative differences.

Zhang et al., (2016) [22] found that the convergence time of a DNN increases as the label corruption ratio increases. Arpit et al., (2017) [1] concluded that DNNs do not use brute-force memorization to fit real datasets but extract patterns in the data based on experimental findings in the dataset of MNIST and CIFAR-10. Arpit et al., (2017) [1] suggests that DNNs may learn simple and general patterns first in the real data. Xu et al., (2018) [20] found similar results as the following. With the simple visualization of low-dimensional functions on Fourier domain, Xu et al., (2018) [20] found F-Principle for low-frequency dominant functions empirically. However, F-Principle does not apply to pure noise data [20].

To theoretically understand the above empirically findings, we first note that the real data in Fourier domain is usually low-frequency dominant [5] while pure noise data often does not have clear dominant frequency components, for example, white noise. For low-frequency dominant functions, DNNs can quickly capture the low-frequency components. However, for pure noise data, since the high-frequency components is also important, that is, $A(\omega)$ in Eq. (3.13) could be large for a large ω, the priority of low-frequency during the training can be relatively small. During the training, the gradients of low frequency and high frequency can affect each other significantly. Thus, the training processes for real data and pure noise data are often very different. Therefore, along with the analysis in *Results*, F-Principle thus can well apply to low-frequency dominant functions but not pure noise data [1, 20]. Since the DNN needs to capture more large-amplitude higher-frequency components when it is fitting pure noise data, it often requires longer convergence time [22].

Memorization vs. generalization. Traditional learning theory—which restricts capacity (e.g., VC dimension [18]) to achieve good generalization—cannot explain how DNNs can have large capacity to memorize random labeled dataset [22], but still possess good generalization in real dataset. As other study has shown that sufficient large-scale DNNs can potentially approximate any function [3]. In this work, our theoretical framework further resolves this puzzle by showing that both the DNN structure and the training dataset affect the effective capacity of DNNs. In the Fourier domain, high-frequency components can increase the DNN capacity and complexity. However, with small initialization, DNNs invoke frequency components from low to high to fit training data and keep other extra-higher frequency components, which are beyond the effective frequency range of training data, small. Thus, DNN's learned capacity and complexity are determined by the training data, however, they still have the potential of large capacity. This effect raised from individual dataset is consistent with the speculation from Kawaguchi et al., (2017) [8] that suggests the generalization ability of deep learning is affected different datasets.

Limitations. i) The theoretical framework in its current form could not analyze how different DNN structures, such as convolutional neural networks, affect the detailed training and generalization ability of DNNs. We believe that to consider the effect of DNN structure, we need to consider more properties of dataset in addition to that its power decays as the frequency increases. ii), this qualitative framework cannot analyze the difference between the number of layers and the width of layers. To this end, we need an exact mathematical form for the DFT of the output of the DNN with multiple hidden layers, which will be left for our future work. iii) this theoretical framework cannot analyze the DNN behavior around local minima/saddle points during training.

Acknowledgments. The author wants to thank Wei Dai, Qiu Yang, Shixiao Jiang for critically proofreading the manuscript.

Appendix A. Fourier transform of sigmoid function[4] **.** The sigmoid function is

$$\sigma(x) = \frac{e^x}{e^x+1}, \quad x \in \mathrm{R}.$$

The Fourier transform [5] is

$$F[\sigma(x)](\omega) = \int_{-\infty}^{\infty} \sigma(x) e^{-i\omega x} \mathrm{d}x. \tag{A.1}$$

$$F[\sigma(x)](\omega) = \int_{-\infty}^{0} + \int_{0}^{\infty} \sigma(x) e^{-i\omega x} \mathrm{d}x.$$

Consider the case $\omega \neq 0$. For the positive part,

$$\int_0^{\infty} \sigma(x) e^{-i\omega x} \mathrm{d}x = \int_0^{\infty} \frac{e^{-i\omega x}}{e^{-x}+1} \mathrm{d}x.$$

Since

$$\frac{1}{e^{-x}+1} = \sum_{k=0}^{\infty} (-1)^k e^{-kx},$$

we have

$$\begin{aligned}\int_0^{\infty} \sum_{k=0}^{\infty} (-1)^k e^{-kx} e^{-i\omega x} \mathrm{d}x &= \sum_{k=0}^{\infty} (-1)^k \int_0^{\infty} e^{-(k+i\omega)x} \mathrm{d}x \\ &= \sum_{k=0}^{\infty} (-1)^k \frac{1}{k+i\omega}.\end{aligned}$$

For the negative part,

$$\int_{-\infty}^{0} \sigma(x) e^{-i\omega x} \mathrm{d}x = \int_{-\infty}^{0} \frac{e^{(1-i\omega)x}}{e^x+1} \mathrm{d}x.$$

[4]The mathematical calculation in this section is inspired by https://math.stackexchange.com/questions/2569814/fourier-transform-of-sigmoid-function.

[5]Without loss of generality, the constant factor that is related to the definition of corresponding Fourier Transform is ignored here. ω is the product of 2π and the inverse of period.

$$\frac{1}{e^x+1}=\sum_{k=0}^{\infty}(-1)^k e^{kx},$$

$$\begin{aligned}\int_{-\infty}^{0}\sigma(x)e^{-i\omega x}\mathrm{d}x &= \sum_{k=0}^{\infty}(-1)^k\int_{-\infty}^{0}e^{(k+1-i\omega)x}\mathrm{d}x\\ &= \sum_{k=0}^{\infty}\frac{(-1)^k}{k+1-i\omega},\end{aligned}$$

then,

$$\begin{aligned}F[\sigma(x)](\omega) &= \sum_{k=0}^{\infty}(-1)^k\frac{1}{k+i\omega}+\sum_{k=0}^{\infty}\frac{(-1)^k}{k+1-i\omega}\\ &= \Phi(-1,1,i\omega)+\Phi(-1,1,1-i\omega)\\ &= \frac{1}{2}i\pi\tanh(\frac{\pi\omega}{2})-\frac{1}{2}i\pi\coth(\frac{\pi\omega}{2})\\ &= -i\pi\mathrm{csch}(\pi\omega),\end{aligned}$$

where Φ is Lerch Phi function,

$$\begin{aligned}\tanh(x) &= \frac{e^x-e^{-x}}{e^x+e^{-x}},\\ \coth(x) &= \frac{e^x+e^{-x}}{e^x-e^{-x}},\\ \mathrm{csch}(x) &= \frac{2}{e^x-e^{-x}}.\end{aligned}$$

Since

$$F[\sigma(ax+b)](\omega)=\frac{e^{ib\omega/a}}{|a|}F[\sigma(x)](\frac{\omega}{a}),$$

we have

$$F[\sigma(ax+b)](\omega)=-i\pi\mathrm{csch}(\frac{\pi\omega}{a})\frac{e^{ib\omega/a}}{|a|},$$

that is,

(A.2) $$F[\sigma(ax+b)](\omega)=-\frac{i\pi}{|a|}\exp\left(ib\omega/a\right)\frac{2}{\exp(\pi\omega/a)-\exp(-\pi\omega/a)}.$$

Then, when $|\pi\omega/a|$ is large, without loss of generality, we take $\pi\omega/a\gg 0$,

(A.3) $$F[\sigma(ax+b)](\omega)\approx-\frac{2i\pi}{|a|}\exp\left(i\frac{b\omega}{a}\right)\exp(-\pi\omega/a).$$

When $\omega=0$, in Eq. (A.1), $F[\sigma(ax+b)](\omega)$ is infinite. However, we only consider a finite range in practice. Thus, the infinity of $\omega=0$ in Eq. (A.2) is not an issue in practice.

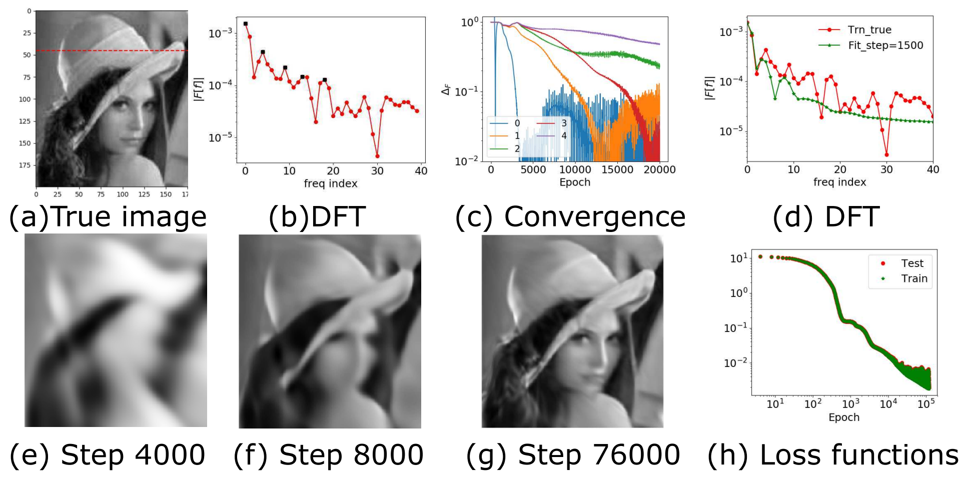


Figure 5: Convergence from low frequency to high frequency for a natural image, Lena. The training data are all pixels whose horizontal indexes are odd. (a) True image. (b) $|F[f]|$ of the red dashed pixels in (a) as a function of frequency index with important peaks marked by black dots. (c) Δ_F at different recording steps for different selected frequency peaks in (b). (d) $|F[f]|$ and $|F[T]|$ at the recording step 1500. (e-g) DNN outputs of all pixels at different steps. (h) Loss functions. We use a DNN with width 500-400-300-200-100. We train the DNN with the full batch and learning rate 2×10^{-5}. We initialize DNN parameters by Gaussian distribution with mean 0, and the standard deviation for weights and bias terms are 0.08 and 3, respectively.

Appendix B. Fitting high-frequency dominant functions . We explain the training process of DNN's fitting of a high-frequency dominant function, which is beyond the F-Principle. Similarly, for small initialization, since Δ_F is independent of the amplitude at the beginning, the low-frequency components will converge earlier, say, ω_L. The frequency of the main contributor to the descent amount would be higher ones, say, ω_H. Since $A(\omega_H)$—the amplitude of the difference between the DNN output and the target function at ω_H—is much lager than $|F[f](\omega_L)|$ for the high-frequency dominant function, from Eq. (5.2), the gradient caused by the frequency component at ω_H could cause a non-ignorable deviation of DNN output from $|F[f](\omega_L)|$ at ω_L. With the training by gradient descent, $A(\omega_L)$ deviates from zero and $A(\omega_H)$ decreases. $A(\omega)\exp(-|\pi\omega/a_j|)$ in Eq. (3.13) would lead to that low-frequency has important contribution to the total descent amount again; thus, we can observe the low-frequency component would once again converge. As the training goes on, $A(\omega_H)$ decreases; thus, the deviation of the DNN output from the target function at frequency ω_L decreases. Therefore, during the training, $\Delta_F(\omega)$ oscillates and decreases its oscillation amplitude. The very low-frequency components would converge after a small deviation. Therefore, we can observe that $\Delta_F(\omega)$ for very low-frequency components oscillates more with smaller amplitudes.

For example, we construct a function $g(x)$ as follows. First, we evenly sample 40 points from $[-1,1]$ for $f(x)=x$. Second, we perform DFT for $f(x)$, that is, $F[f](\omega)$. Third, we flip $F[f](\omega)$ to derive $g(x)$ as shown in Fig.6a and Fig.6b. In Fig. 6c, except for the highest peak (the 19th frequency), $\Delta_F(\omega)$ of other frequency components oscillate and decrease their amplitudes. For a very

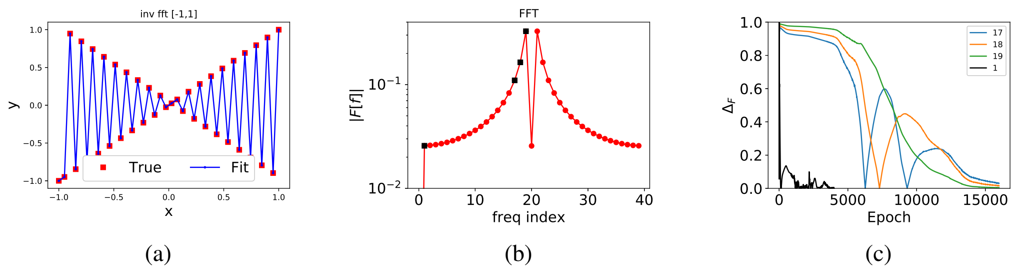


Figure 6: Frequency domain analysis for high-frequency dominant function $g(x)$, whose DFT is obtained by flipping the DFT of $y = x$ with 40 evenly sampled points from $[-1, 1]$. (a) red dots is for $g(x)$, blue squares (connected by blue lines) are for the DNN output at the end of training. (b) FFT of $g(x)$. (c) Δ_F at different recording steps for different frequency peaks (different curves). We use a tanh DNN with width: 200-200-200-200-100. The learning rate is 2×10^{-5} with the full batch. DNN parameters are initialized by Gaussian distribution with mean 0 and standard deviation 0.1.

low-frequency components, such as the first frequency component in Fig. 6c, its $\Delta_F(\omega)$ oscillates more, and its amplitude is small compared with other frequencies in Fig. 6c.

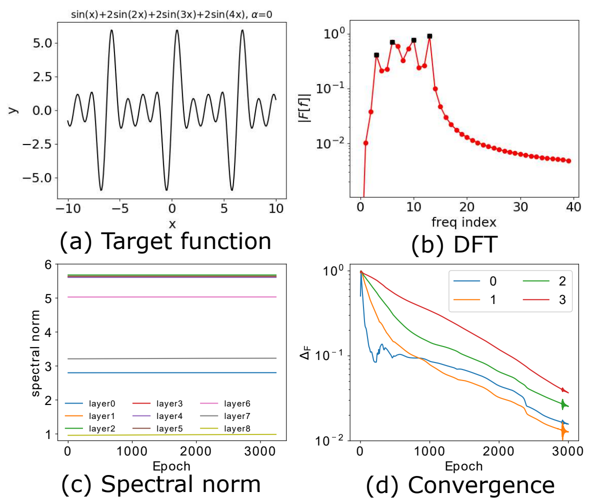


Figure 7: Convergence from low frequency to high frequency for a 1-d function while the spectral norm almost does not change. (a) The target function. (b) $|F[f]|$ (red solid line) as a function of frequency index with important peaks marked by black dots. (c) Spectral norm of all weights. As the same as Rahaman et al., (2018) [16], for matrix-valued weights, their spectral norm was computed by evaluating the eigenvalue of the eigenvector. For vector-valued weights, we simply use the L_2 norm. (d) Δ_F at different recording steps for different selected frequency peaks in (b). The training data are evenly sampled in $[-10, 10]$ with sample size 120. We use a DNN with width: 800-800-800-800-800-800-500-100. We train the DNN with the full batch and learning rate 2×10^{-6}. We initialize DNN parameters by Gaussian distribution with mean 0 and standard deviation 0.1.

REFERENCES

[1] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al., *A closer look at memorization in deep networks*, arXiv preprint arXiv:1706.05394, (2017).

[2] O. Bousquet and A. Elisseeff, *Stability and generalization*, Journal of machine learning research, 2 (2002), pp. 499–526.

[3] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems, 2 (1989), pp. 303–314.

[4] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, *Sharp minima can generalize for deep nets*, arXiv preprint arXiv:1703.04933, (2017).

[5] D. W. Dong and J. J. Atick, *Statistics of natural time-varying images*, Network: Computation in Neural Systems, 6 (1995), pp. 345–358.

[6] M. Hardt, B. Recht, and Y. Singer, *Train faster, generalize better: Stability of stochastic gradient descent*, arXiv preprint arXiv:1509.01240, (2015).

[7] S. Hochreiter and J. Schmidhuber, *Simplifying neural nets by discovering flat minima*, in Advances in neural information processing systems, 1995, pp. 529–536.

[8] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, *Generalization in deep learning*, arXiv preprint arXiv:1710.05468, (2017).

[9] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, arXiv preprint arXiv:1609.04836, (2016).

[10] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[11] Y. LeCun, *The mnist database of handwritten digits*, http://yann. lecun. com/exdb/mnist/, (1998).
[12] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, nature, 521 (2015), p. 436.
[13] M. Mishali and Y. C. Eldar, *Blind multiband signal reconstruction: Compressed sensing for analog signals*, IEEE Transactions on Signal Processing, 57 (2009), pp. 993–1009.
[14] V. Nagarajan and J. Z. Kolter, *Generalization in deep networks: The role of distance from initialization*, in NIPS workshop on Deep Learning: Bridging Theory and Practice, 2017.
[15] T. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar, *Theory of deep learning iii: the non-overfitting puzzle*, tech. report, Technical report, CBMM memo 073, 2018.
[16] N. Rahaman, D. Arpit, A. Baratin, F. Draxler, M. Lin, F. A. Hamprecht, Y. Bengio, and A. Courville, *On the spectral bias of deep neural networks*, arXiv preprint arXiv:1806.08734, (2018).
[17] C. E. Shannon, *Communication in the presence of noise*, Proceedings of the IRE, 37 (1949), pp. 10–21.
[18] V. N. Vapnik, *An overview of statistical learning theory*, IEEE transactions on neural networks, 10 (1999), pp. 988–999.
[19] L. Wu, Z. Zhu, and W. E, *Towards understanding generalization of deep learning: Perspective of loss landscapes*, arXiv preprint arXiv:1706.10239, (2017).
[20] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, *Training behavior of deep neural network in frequency domain*, arXiv preprint arXiv:1807.01251, (2018).
[21] J. Yen, *On nonuniform sampling of bandwidth-limited signals*, IRE Transactions on circuit theory, 3 (1956), pp. 251–257.
[22] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, *Understanding deep learning requires rethinking generalization*, arXiv preprint arXiv:1611.03530, (2016).