Defects of Convolutional Decoder Networks in Frequency Representation

Ling Tang [*1] **Wen Shen** [*1] **Zhanpeng Zhou** [1] **Yuefeng Chen** [2] **Quanshi Zhang** [1 3]

Abstract

In this paper, we prove the representation defects of a cascaded convolutional decoder[1] network, considering the capacity of representing different frequency components of an input sample. We conduct the discrete Fourier transform on each channel of the feature map in an intermediate layer of the decoder network. Then, we extend the 2D circular convolution theorem to represent the forward and backward propagations through convolutional layers in the frequency domain. Based on this, we prove three defects in representing feature spectrums. First, we prove that the convolution operation, the zero-padding operation, and a set of other settings all make a convolutional decoder network more likely to weaken high-frequency components. Second, we prove that the upsampling operation generates a feature spectrum, in which strong signals repetitively appear at certain frequencies. Third, we prove that if the frequency components in the input sample and frequency components in the target output for regression have a small shift, then the decoder usually cannot be effectively learned.

1. Introduction

In this study, we investigate the representation defect of a cascaded convolutional decoder[1] in generating features at different frequencies. That is, when we apply the discrete Fourier transform (DFT) to each channel of the feature map or the input sample, we try to prove which frequency components of each input channel are usually strengthened/weakened by the network.

We extend the 2D circular convolution theorem to reformulate the forward propagation through multiple convolutional layers in the frequency domain. We find that both the forward propagation and the backward propagation in a convolutional network can be represented as the matrix multiplication on spectrums of the feature. Specifically, we mainly analyze a convolutional decoder, which only contains convolution operations without changing the size of feature maps. Then, based on the propagation in the frequency domain, we prove the following conclusions.

• *Problem in representing high-frequency components.* We prove that both the convolution operation and the zero-padding operation make a cascaded convolutional decoder network more likely to weaken the high-frequency components of the input sample, if the convolution operation with a padding operation does not change the size of the feature map in a channel, as shown in Figure 1(a). Besides, we also prove that the following three conditions further strengthen the above representation problem, including (1) a deep network architecture; (2) a small convolutional kernel size; and (3) a large absolute value of the mean value of convolutional weights.

• *Problem in mistakenly repeating certain frequencies.* We find that the upsampling operation makes a cascaded convolutional decoder network generate a feature spectrum, in which strong signals repetitively appear at certain frequencies, as shown in Figure 1(b).

• *Problem in fitting specific frequency components.* More crucially, we discover and prove that it is usually difficult to train an auto-encoder to fit the target image, if salient frequency components of the target output and those of the input have a small shift in the spectrum. Considering the continuous success of the auto-encoder in recent years, such a phenomenon is quite contrary to intuition. As Figure 1(c) shows, a smaller shift between the input spectrum and the output spectrum usually leads to a higher difficulty in training the auto-encoder.

The above three problems just explain general trends towards generic problems of neural networks with convolution, zero-padding, and upsampling operations, instead of deriving a deterministic property of a specific network.

Although most conclusions are derived by ignoring ReLU

[*]Equal contribution [1]Shanghai Jiao Tong University. [2]Alibaba Group. [3]Quanshi Zhang is the corresponding author. He is with the Department of Computer Science and Engineering, the John Hopcroft Center, at the Shanghai Jiao Tong University, China. Correspondence to: Quanshi Zhang <zqs1022@sjtu.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]Here, the decoder represents a typical network, whose feature map size is non-decreasing during the forward propagation.

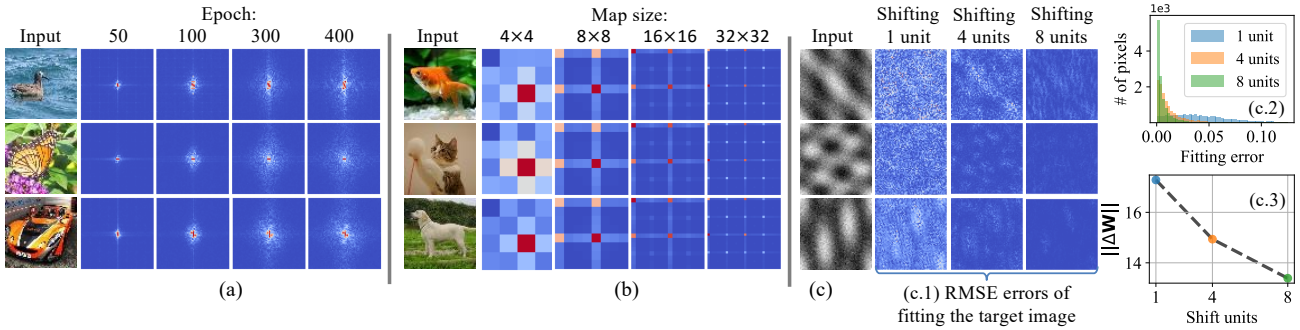


Figure 1. Three representation problems with a cascaded convolutional decoder network. (a) The convolution operation and the zero-padding operation make the decoder usually learn low-frequency components first and then gradually learn higher frequencies. (b) For cascaded upconvolutional layers, the upsampling operation in the decoder repeats strong frequency components of the input to generate spectrums of upper layers. High-frequency components in magnitude maps in (b) are also weakened by the convolution operation after upsampling. We visualize the magnitude map of the feature spectrum, which is averaged over all channels. For clarity, we move low frequencies to the center of the spectrum map, and move high frequencies to the corners of the spectrum map. (c) The auto-encoder usually cannot be trained to fit the target output, whose specific frequency components have a small shift from the spectrum of the input image. We visualize the heatmap of fitting errors (c.1), *i.e.*, the root mean squared error (RMSE), visualize the histogram of fitting errors over different pixels (c.2), and report the learning difficulty $\|\Delta \mathbf{W}\|$ (c.3). Here, results in (c.2) and (c.3) are averaged over different DNNs. Note that for magnitude maps in (a), we set the magnitude of the fundamental frequency to be the same with the magnitude of the second significant frequency.

operations in the decoder, we have conducted experiments, which have successfully verified such defects in different multi-layer decoder networks with ReLU layers. This proves the trustworthiness of our theorems. Note that we have not derived the property of max-pooling operations, so in this paper, it is difficult to extend such findings to neural networks for image classification.

Discussions on two types of frequencies. People usually analyze feature representations of a network considering two types of frequencies. (Xu et al., 2019a; Rahaman et al., 2019) took the landscape of the loss function on all input samples as the time domain to analyze the frequency in the sample space. In comparison, we focus on the second type of frequency, *i.e.*, we apply DFT to each channel of the intermediate-layer feature of a convolutional decoder and analyze defects in representing specific frequencies.

2. Related work

Although few previous studies directly prove a DNN's defects from the perspective of representing specific feature components, we still make a survey on research on the representation capacity of a DNN.

Some studies focused on a specific frequency that took the landscape of the loss function on all input samples as the time domain (Xu et al., 2019b; Rahaman et al., 2019; Xu et al., 2019a; Luo et al., 2019). Based on such a specific frequency, they observed and proved a phenomenon namely Frequency Principle (F-Principle) that a DNN first quickly learned low-frequency components, and then relatively slowly learned the high-frequency ones. Ma *et al.* (2020) further explored the boundary of the F-Principle, beyond which the F-Principle did not hold anymore. Besides, Lin *et al.* (2019) empirically proposed to smooth out high-frequency components to improve the adversarial robustness. **In comparison, we focus on a fully different type of frequency**, *i.e.*, the frequency *w.r.t.* the DFT on an input image or a feature map.

In this direction, previous studies mainly experimentally analyzed the relationship between the learning of different frequencies and the robustness of a DNN. Yin *et al.* (2019) conducted a lot of experiments to analyze the robustness of a DNN *w.r.t.* different frequencies of the image. They discovered that both adversarial training and Gaussian data augmentation improved the DNN's robustness to higher frequencies. Wang *et al.* (2020) empirically proposed to remove high-frequency components of convolutional weights to improve the adversarial robustness. In comparison, we theoretically prove representation defects of DNNs in the frequency domain.

In fact, many studies explained the representation capacity of a DNN in the **time domain**. The information bottleneck hypothesis (Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017; Wolchover & Reading, 2017; Amjad & Geiger, 2019) showed that the learning process of DNNs was to retain the task-relevant input information and discarded the task-irrelevant input information. The lottery ticket hypothesis (Frankle & Carbin, 2018) showed that some initial parameters of DNNs inherently contributed more to the network output. The double-descent phe-

nomenon (Nakkiran et al., 2019; Reinhard & Fatih, 2020) described the specific training process of DNNs that the loss first declined, then rose, and then declined again. DNNs with batch normalization were sometimes conflicted with the weight decay (Van Laarhoven, 2017; Li et al., 2020). DNNs were difficult to encode interactions between an intermediate number of input variables (Deng et al., 2022).

3. Propagation in the frequency domain

Preliminary 1, convolution operation. Given a convolutional layer, let $\mathbf{W}^{[ker=1]}$, $\mathbf{W}^{[ker=2]}, \ldots,$ $\mathbf{W}^{[ker=D]}$ denote D convolutional kernels of this layer, and let $b^{[ker=1]}, b^{[ker=2]}, \ldots, b^{[ker=D]} \in \mathbb{R}$ denote D bias terms. Each d-th kernel $\mathbf{W}^{[ker=d]} \in \mathbb{R}^{C\times K\times K}$ is of the kernel size $K \times K$, and C denotes the channel number. Accordingly, we apply these kernels on a feature $\mathbf{F} \in \mathbb{R}^{C\times M\times N}$ with C channels, and obtain the output feature $\widetilde{\mathbf{F}} \in \mathbb{R}^{D\times M'\times N'}$, as follows.

$$\widetilde{\mathbf{F}} = Conv(\mathbf{F}), \quad s.t.\ \forall d,\ \widetilde{\mathbf{F}}^{(d)} = \mathbf{W}^{[ker=d]} \otimes \mathbf{F} + b^{[ker=d]}\mathbf{1}_{M'\times N'}, \tag{1}$$

where $\widetilde{\mathbf{F}}^{(d)} \in \mathbb{R}^{M'\times N'}$ denotes the feature map of the d-th channel. $\otimes$ denotes the convolution operation. $\mathbf{1}_{M'\times N'} \in \mathbb{R}^{M'\times N'}$ is an all-ones matrix.

Preliminary 2, discrete Fourier transform. Given the c-th channel of the feature $\mathbf{F} \in \mathbb{R}^{C\times M\times N}$, *i.e.*, $F^{(c)} \in \mathbb{R}^{M\times N}$, we use the discrete Fourier transform (DFT) (Sundararajan, 2001) to compute the frequency spectrum of this channel, which is termed $G^{(c)} \in \mathbb{C}^{M\times N}$, as follows. $\mathbb{C}$ denotes the algebra of complex numbers.

$$\forall u, v, \quad G^{(c)}_{uv} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F^{(c)}_{mn} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi}. \tag{2}$$

Each frequency component at the frequency $[u, v]$ is represented as a complex number, *i.e.*, $G^{(c)}_{uv} \in \mathbb{C}$. Let $\mathbf{G} = [G^{(1)}, \ldots, G^{(C)}] \in \mathbb{C}^{C\times M\times N}$ denote the tensor of frequency spectrums of the C channels of $\mathbf{F}$. We take the C-dimensional vector at the frequency $[u, v]$ of the tensor $\mathbf{G}$, *i.e.*, $\mathbf{g}^{(uv)} = [G^{(1)}_{uv}, G^{(2)}_{uv}, \ldots, G^{(C)}_{uv}]^\top \in \mathbb{C}^C$, to represent the frequency component $[u, v]$ of the feature $\mathbf{F}$. Frequency components closed to $[0, 0]$, $[0, N-1]$, $[M-1, 0]$, or $[M-1, N-1]$ represent low-frequency signals, whereas frequency components closed to $[\frac{M}{2}, \frac{N}{2}]$ represent high-frequency signals.

3.1. Propagation in frequency

In this section, we extend the 2D circular convolution theorem (Jain, 1989) to represent the forward propagation and the back-propagation in a cascaded convolutional network.

Assumption 3.1. *Let us follow the setting in the 2D circular convolution theorem (Jain, 1989), which adds the circular padding operation assumption (Jain, 1989) to the convolution operation. The convolution operation is conducted with a circular padding and with a stride size of 1, so as to avoid the convolution changing the size of the feature map. The circular padding is used to extend the last row and the last column of the feature map in each channel.*

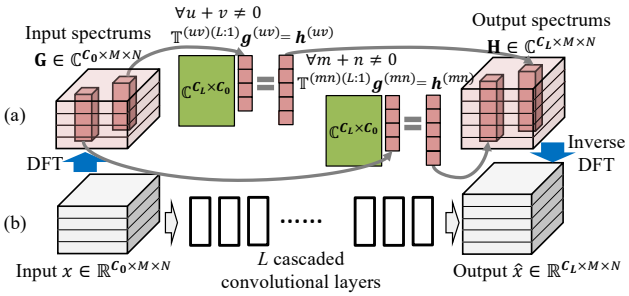


Figure 2. Forward propagation in the frequency domain (a) and forward propagation in the time domain (b). The cascaded convolution operations on input x are essentially equivalent to matrix multiplication on spectrums $\mathbf{G}$ of the input.

Theorem 3.2. *(Proof in Appendix A.1). According to Assumption 3.1, the output feature $\widetilde{\mathbf{F}} \in \mathbb{R}^{D\times M\times N}$ has the same size as the input feature. Let $\boldsymbol{H} = [H^{(1)}, H^{(2)}, \ldots, H^{(D)}] \in \mathbb{C}^{D\times M\times N}$ denote a tensor consisting of D spectrums corresponding to the D channels of $\widetilde{\mathbf{F}}$. Then, $\boldsymbol{H}$ can be computed as follows.*

$$\boldsymbol{h}^{(uv)} = T^{(uv)}\boldsymbol{g}^{(uv)} + \delta_{uv}MN\boldsymbol{b} \quad s.t.\ \delta_{uv} = \begin{cases} 1, & u=v=0 \\ 0, & otherwise \end{cases} \tag{3}$$

where $\boldsymbol{h}^{(uv)} = [H^{(1)}_{uv}, H^{(2)}_{uv}, \ldots, H^{(D)}_{uv}]^\top \in \mathbb{C}^D$ denotes a column at the frequency $[u, v]$ in the tensor $\boldsymbol{H}$; $T^{(uv)} \in \mathbb{C}^{D\times C}$ is a matrix of complex numbers and is exclusively determined by convolutional kernels $\mathbf{W}^{[ker=1]}, \mathbf{W}^{[ker=2]}, \ldots, \mathbf{W}^{[ker=D]}$, $T^{(uv)}_{dc} = \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W^{[ker=d]}_{cts} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi}$; $\boldsymbol{b} = [b^{(1)}, b^{(2)}, \ldots, b^{(D)}]^\top \in \mathbb{R}^D$ denotes the vector of bias terms.

To simplify the further proof, we temporarily investigate the spectrum propagation of a network with L cascaded convolutional layers, but does not contain activation functions. Let us first discuss the trustworthiness of such a simplification. We have conducted experiments to show that all our findings in all theorems can also well explain the properties of an ordinary cascaded convolutional network with ReLU layers. As shown in Figure 3, for a network with ReLU layers, although the value derived from our theory was not exactly the same as the real value, experimental results still verified the conclusions of our theory. More crucially, experiments in Figures 1, 3, 4, and 5 were all conducted on ReLU networks.

Let a convolutional network contain L cascaded convolutional layers. Each l-th layer contains C_l convolutional kernels, $\mathbf{W}^{(l)[ker=1]}, \mathbf{W}^{(l)[ker=2]}, \ldots, \mathbf{W}^{(l)[ker=C_l]} \in \mathbb{R}^{C_{l-1}\times K\times K}$, with C_l bias terms $b^{(l,1)}, b^{(l,2)}, \ldots, b^{(l,C_l)} \in \mathbb{R}$. Let $x \in \mathbb{R}^{C_0\times M\times N}$ denote the input sample. The network generates the output sample $\widehat{x} = \text{net}(x) \in \mathbb{R}^{C_L\times M\times N}$. Then, we derive the forward propagation of spectrums of x to spectrums of $\widehat{x}$ in the frequency domain, as follows.

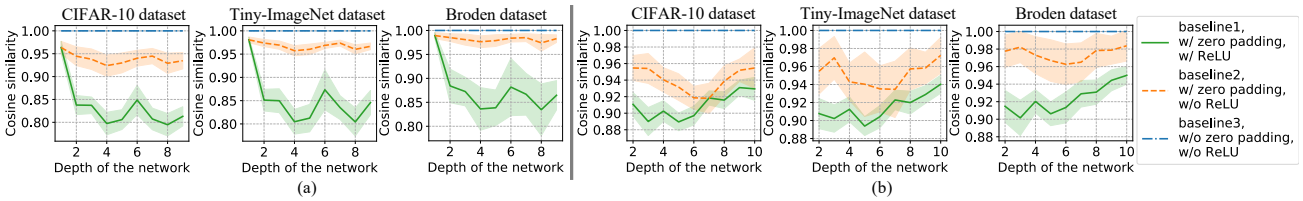


Figure 3. (a) Fitness between the derived feature spectrums $\mathbf{H}$ in Corollary 3.3 and the real feature spectrums $\mathbf{H}^*$ measured in a real DNN. (b) Fitness between the derived change of $T^{(l,uv)}$ in Corollary 3.4 and the real $T^{(l,uv)}$ measured in a real DNN. The shaded area represents the standard deviation.

Corollary 3.3. *(Proof in Appendix A.2) Let* $\boldsymbol{G} = [G^{(1)}, G^{(2)}, \dots, G^{(C_0)}] \in \mathbb{C}^{C_0 \times M \times N}$ *denote frequency spectrums of the* C_0 *channels of the input* x*. Then, based on Assumption 3.1, spectrums of the image* $\widehat{x}$ *generated by* L *cascaded convolutional layers, i.e.,* $\boldsymbol{H} = [H^{(1)}, H^{(2)}, \dots, H^{(C_L)}] \in \mathbb{C}^{C_L \times M \times N}$*, are given as*

$$\boldsymbol{h}^{(uv)} = \mathbb{T}^{(uv)(L:1)} \boldsymbol{g}^{(uv)} + \delta_{uv} \boldsymbol{\beta} \tag{4}$$

where $\boldsymbol{g}^{(uv)} = [G_{uv}^{(1)}, G_{uv}^{(2)}, \dots, G_{uv}^{(C_0)}]^\top \in \mathbb{C}^{C_0}$ *and* $\boldsymbol{h}^{(uv)} = [H_{uv}^{(1)}, H_{uv}^{(2)}, \dots, H_{uv}^{(C_L)}]^\top \in \mathbb{C}^{C_L}$ *denote vectors at the frequency* $[u, v]$ *in tensors* $\boldsymbol{G}$ *and* $\boldsymbol{H}$*, respectively.* $\mathbb{T}^{(uv)(L:1)} = T^{(L,uv)} \cdots T^{(2,uv)} T^{(1,uv)} \in \mathbb{C}^{C_L \times C_0}$*.* $\boldsymbol{\beta} = MN\big(\boldsymbol{b}^{(L)} + \sum_{j=2}^{L} \mathbb{T}^{(00)(L:j)} \boldsymbol{b}^{(j-1)}\big) \in \mathbb{C}^{C_L}$*.* $\boldsymbol{b}^{(l)} = [b^{(l,1)}, b^{(l,2)}, \dots, b^{(l,C_l)}]^\top \in \mathbb{R}^{C_l}$ *denotes the vector of bias terms of* C_l *convolutional kernels in the* l*-th layer.*

Understanding the cascaded convolution operations in the frequency domain. Corollary 3.3 means that conducting multiple cascaded convolution operations on an input x is essentially equivalent to conducting matrix multiplication on spectrums of x. As Figure 2 shows, for all frequencies except for the fundamental frequency, we have the output spectrum $\mathbf{h}^{(uv)} = \mathbb{T}^{(uv)(L:1)} \mathbf{g}^{(uv)}$.

Besides, the learning of parameters $\mathbf{W}^{(l)}$ affects the matrix $T^{(l,uv)}$. Therefore, we further reformulate the change of $T^{(l,uv)}$ during the learning process, as follows.

Corollary 3.4. *(Proof in Appendix A.3) Based on Assumption 3.1, the change of each frequency component* $T^{(l,uv)}$ *during the learning process is reformulated, as follows.*

$$(\Delta T^{(l,uv)})^\top = -\eta MN \sum_{u'=0}^{M-1} \sum_{v'=0}^{N-1} \chi_{u'v'uv} \Big(\overline{\mathbb{T}}^{(u'v')(l-1:1)} \overline{\boldsymbol{g}}^{(u'v')} + \delta_{u'v'} \overline{\boldsymbol{\beta}}' \Big) \frac{\partial Loss}{\partial (\overline{\boldsymbol{h}}^{(u'v')})^\top} \overline{\mathbb{T}}^{(u'v')(L:l+1)}; \tag{5}$$

where $\chi_{u'v'uv} = \frac{1}{MN} \frac{\sin(\frac{K(u-u')\pi}{M})}{\sin(\frac{(u-u')\pi}{M})} \frac{\sin(\frac{K(v-v')\pi}{N})}{\sin(\frac{(v-v')\pi}{N})} \cdot e^{i(\frac{(K-1)(u-u')}{M} + \frac{(K-1)(v-v')}{N})\pi} \in \mathbb{C}$ *is a coefficient;* $\mathbb{T}^{(u'v')(l-1:1)} = T^{(l-1,u'v')} \cdots T^{(2,u'v')} T^{(1,u'v')} \in \mathbb{C}^{C_{l-1} \times C_0}$*;* $\mathbb{T}^{(u'v')(L:l+1)} = T^{(L,u'v')} \cdots T^{(l+1,u'v')} \in \mathbb{C}^{C_L \times C_l}$*;* $\boldsymbol{\beta}' = MN\big(\boldsymbol{b}^{(l-1)} + \sum_{j=2}^{l-1} \mathbb{T}^{(00)(l-1:j)} \boldsymbol{b}^{(j-1)}\big) \in \mathbb{C}^{C_{l-1}}$*;* $\overline{\boldsymbol{g}}^{(u'v')}$ *denotes the conjugate of* $\boldsymbol{g}^{(u'v')}$*;* η *is the learning rate.*

3.2. Experimental verification of Corollaries 3.3 and 3.4

To verify the correctness of Corollary 3.3, we computed the similarity between real spectrums $\mathbf{H}^* = [H^{*(1)}, H^{*(2)}, \cdots]$ measured by applying the DFT to the real network output, and spectrums $\mathbf{H} = [H^{(1)}, H^{(2)}, \cdots]$ derived in Corollary 3.3. Specifically, we measured the cosine similarity $sim(\mathbf{H}^*, \mathbf{H}) = \mathbb{E}_c[\mathbf{cos}(\mathbf{vec}(mag(H^{*(c)})), \mathbf{vec}(mag(H^{(c)})))]$, where $\mathbf{vec}(\cdot)$ represented the vectorization of a matrix, and $mag(\cdot)$ transferred a complex-valued matrix to a real-valued magnitude matrix[2].

To this end, we constructed the following three baseline networks to verify whether Corollary 3.3 derived from specific assumptions could also objectively reflect real forward propagations in real neural networks. The first baseline network contained 10 convolutional layers. Each convolutional layer applied zero-paddings and was followed by an ReLU layer. Each convolutional layer contained 16 convolutional kernels (kernel size was 3×3) with 16 bias terms. We set the stride size of the convolution operation to 1. The second baseline network was constructed by removing all ReLU layers from the first baseline network, which was closer to the assumption in Corollary 3.3. The third baseline network was revised from the second baseline network by replacing all zero-paddings with circular paddings. The third baseline network followed the exact assumption in Corollary 3.3.

Figure 3(a) reports $sim(\mathbf{H}^*, \mathbf{H})$ that was measured on spectrums in different layers and averaged over all samples. The similarity between real spectrums and derived spectrums was large for all the three baseline networks, which verified Corollary 3.3. Note that the cosine similarity was computed based on high-dimensional vectors with as many as 32^2, 64^2 or 224^2 dimensions (determined by the dataset). For such high-dimensional vectors, a similarity greater than 0.8 was already significant enough to verify the practicality of our theory[3].

Besides, in order to verify Corollary 3.4, we also measured the similarity between the real change of $T^{(l,uv)}$

[2]The function $B = mag(A)$ returns a matrix, where each element $B_{ij} \in \mathbb{R}$ represents the magnitude of $A_{ij} \in \mathbb{C}$.

[3]Please see Appendix C.6 for the curse of dimension.

computed by measuring real network parameters, termed $\Delta^* T^{(l,uv)}$, and the change of $T^{(l,uv)}$ derived with assumptions in Corollary 3.4, termed $\Delta T^{(l,uv)}$. The similarity was also computed[2] as $sim(\Delta^* T^{(l,uv)}, \Delta T^{(l,uv)}) = \mathbb{E}_c[\textbf{cos}(\textbf{vec}(mag(\Delta^* T^{(l,uv)})), \textbf{vec}(mag(\Delta T^{(l,uv)})))]$. The verification was also conducted on the above three baseline networks. Figure 3(b) reports $\forall l, sim(\Delta^* T^{(l,uv)}, \Delta T^{(l,uv)})$ averaged over all samples. The similarity was greater than 0.88 for all three baseline networks, which was large[3] enough to verify Corollary 3.4.

4. Representation problems

In this section, we aim to prove three defects in the frequency representation with a cascaded convolutional decoder network. Note that unlike previous studies (Xu et al., 2019a; Rahaman et al., 2019) extracting frequent components in the sample space, we focus on a more commonly-used frequency, *i.e.*, applying DFT to each channel of the intermediate-layer feature.

4.1. Effects of the convolution operation

Given an initialized, cascaded, convolutional decoder[1] network with L convolutional layers, let us focus on the behavior of the decoder network in the early epochs of training. We notice that each element in the matrix $T^{(l,uv)}$, *i.e.*, $T^{(l,uv)}_{dc}$, is exclusively determined by the c-th channel of the d-th kenel $W^{(l)[ker=d]}_{c,0:K-1,0:K-1} \in \mathbb{R}^{K\times K}$, according to Theorem 3.2. Because parameters in $W^{(l)}$ in the decoder network are initialized to random noises, we can consider that all elements in $T^{(l,uv)}$ are irrelevant to each other, *i.e.*, $\forall d \neq d', c \neq c', T^{(l,uv)}_{dc}$ is irrelevant to $T^{(l,uv)}_{d'c'}$. Similarly, since different layers' parameters $W^{(l)}$ are irrelevant to each other in the initialized decoder, we can consider that elements in different layers' $T^{(l,uv)}$ are irrelevant to each other, *i.e.*, $\forall l \neq l'$, elements in $T^{(l,uv)}$ and elements in $T^{(l',uv)}$ are irrelevant to each other. Moreover, since the early training of a DNN mainly modifies a few parameters according to the lottery ticket hypothesis (Frankle & Carbin, 2018), we can still assume such irrelevant relationships in early epochs, as follows.

Assumption 4.1. *(Proof in Appendix A.4) We assume that all elements in $T^{(l,uv)}$ are irrelevant to each other, and $\forall l \neq l'$, elements in $T^{(l,uv)}$ and $T^{(l',uv)}$ are irrelevant to each other in early epochs.*

$$\forall d \neq d'; \forall c \neq c', \quad \mathbb{E}_{\mathbf{W}^{(l)}}[T^{(l,uv)}_{dc} T^{(l,uv)}_{d'c'}] = \mathbb{E}_{\mathbf{W}^{(l)}}[T^{(l,uv)}_{dc}]\mathbb{E}_{\mathbf{W}^{(l)}}[T^{(l,uv)}_{d'c'}] \tag{6}$$

$$\forall l, d, c, d', c', \ \mathbb{E}_{\mathbf{W}^{(l)},\ldots,\mathbf{W}^{(1)}}[T^{(l,uv)}_{dc} \mathbb{T}^{(uv)(l-1:1)}_{d'c'}] = \mathbb{E}_{\mathbf{W}^{(l)}}[T^{(l,uv)}_{dc})]\mathbb{E}_{\mathbf{W}^{(l-1)},\ldots,\mathbf{W}^{(1)}}[\mathbb{T}^{(uv)(l-1:1)}_{d'c'}] \tag{7}$$

Besides, according to experimental experience, the mean value of all parameters in $\mathbf{W}^{(l)}$ usually has a small bias during the training process, instead of being exactly zero. Therefore, let us assume that in early epochs, each parameter in $\mathbf{W}^{(l)}$ is sampled from a Gaussian distribution $N(\mu_l, \sigma_l^2)$.

Note that we also experimentally verify that Assumption 4.1 can be also applied to fully trained DNNs, besides DNNs trained after early epochs. Please see Appendix C.7 for details.

According to $\mathbf{h}^{(uv)} = \mathbb{T}^{(uv)(L:1)}\mathbf{g}^{(uv)} + \delta_{uv} MN\mathbf{b}$ in Corollary 3.3, **we investigate the magnitude of $\mathbb{T}^{(uv)(L:1)}$ as an indicator to measure the strength of the network encoding this specific frequency component $\mathbf{g}^{(uv)}$.**

Theorem 4.2. *(Proof in Appendix A.4) Let us focus on the simplest case that each convolutional layer only contains a feature map with a single channel, i.e., $\forall l, C_l = 1$. Based on Assumption 4.1, $\mathbb{T}^{(uv)(L:1)} \in \mathbb{C}$ is computed as $T^{(L,uv)} \cdots T^{(2,uv)} T^{(1,uv)}$, which is the product of L complex numbers. Because each complex number $T^{(l,uv)}$ follows a Gaussian distribution[4], the mean value of $\mathbb{T}^{(uv)(L:1)}$ is $\prod_{l=1}^{L} \mu_l R_{uv} \in \mathbb{C}$, where $R_{uv} = \frac{\sin(\frac{uK\pi}{M})}{\sin(\frac{u\pi}{M})} \frac{\sin(\frac{vK\pi}{N})}{\sin(\frac{v\pi}{N})} e^{i(\frac{(K-1)u}{M} + \frac{(K-1)v}{N})\pi} \in \mathbb{C}$ is a complex coefficient; $0 \leq |R_{uv}| \leq K^2$. The logarithm of the second-order moment is given as $\log SOM(\mathbb{T}^{(uv)(L:1)}) = \sum_{l=1}^{L} \log(|\mu_l R_{uv}|^2 + K^2\sigma_l^2) \in \mathbb{R}$.*

Theorem 4.2 tells us the following five conclusions.

(1) The magnitude of $\mathbb{T}^{(uv)(L:1)}$, which is measured using the second-order moment $SOM(\mathbb{T}^{(uv)(L:1)})$, increases along with the following four terms, including the absolute value of the expectation $|\mu_l|$, the magnitude of the complex coefficient $|R_{uv}|$, the kernel size K, and the variance σ_l^2.

(2) For each frequency component $[u, v]$, the magnitude of $\mathbb{T}^{(uv)(L:1)}$ will exponentially increase along with the depth L of the network. **We can consider that each layer' $T^{(l,uv)}$ has independent effects** $\log(|\mu_l R_{uv}|^2 + K^2\sigma_l^2)$ **on** $\log SOM(\mathbb{T}^{(uv)(L:1)}) = \sum_{l=1}^{L} \log(|\mu_l R_{uv}|^2 + K^2\sigma_l^2)$. We admit that such an conclusion is derived from the second-order moment of $\mathbb{T}^{(uv)(L:1)}$, instead of a deterministic claim for a specific neural network. Nevertheless, according to the Law of Large Numbers, $SOM(\mathbb{T}^{(uv)(L:1)})$ is still a convincing metric to reflect the average significance of $\mathbb{T}^{(uv)(L:1)}$.

For the general case that each convolutional kernel contains more than one channel, *i.e.*, $\forall l, C_l > 1$, the magnitude of $\mathbb{T}^{(uv)(L:1)}$ also approximately exponentially increases along with the network depth with a quite complicated analytic solution. Please see Appendix A.4 for the proof.

[4]The Gaussian distribution of complex numbers has three parameters $\mu \in \mathbb{C}, \sigma^2 \in \mathbb{R}$ and $r \in \mathbb{C}$, which control the mean value, the variance, and the diversity of the phase of the sampled complex number, respectively.

(3) The convolution operation makes a cascaded convolutional decoder network more likely to weaken the high-frequency components of the input sample, if the convolution operation does not change the feature map size. Especially, when the decoder network is deep, such a problem is more significant. See Appendix B.1 for more discussions.

(4) If the expectation μ_l of convolutional weights in each l-th layer has a large absolute value $|\mu_l|$, then the decoder network is less likely to learn high-frequency components. Please see Appendix B.2 for more discussions.

(5) If the convolutional kernel size K is small, then the decoder network is less likely to learn high-frequency components. Please see Appendix B.3 for more discussions.

Experiments in Section 5.1 have verified the above conclusions in the general case that each convolutional layer contains more than one feature map.

4.2. Effects of the zero-padding operation

To simplify the proof, let us consider the following one-side zero-padding. Given each c-th channel $F^{(c)} \in \mathbb{R}^{M\times N}$ of the feature map, the zero-padding puts zero values at the edge of $F^{(c)}$, so as to obtain a new feature $\widetilde{F}^{(c)} \in \mathbb{R}^{M'\times N'}$.

$$\forall m,n,\ \ \widetilde{F}^{(c)}_{mn} = \begin{cases} F^{(c)}_{mn}, & 0\le m<M,\ 0\le n<N \\ 0, & M\le m<M',\ N\le n<N' \end{cases} \tag{8}$$

We have proven that the zero-padding operation boosts magnitudes of low-frequency components of feature spectrums of the feature map, as shown in Theorem 4.3.

Theorem 4.3. *(Proof in Appendix A.5) Let each element in each c-th channel $F^{(c)}$ of the feature map follows the Gaussian distribution $\mathcal{N}(a,\sigma^2)$. $G^{(c)} \in \mathbb{C}^{M\times N}$ denotes the frequency spectrum of $F^{(c)}$, and $H^{(c)} \in \mathbb{C}^{M'\times N'}$ denotes the frequency spectrum of the output feature $\widetilde{F}^{(c)}$ after applying zero-padding on $F^{(c)}$. Then, the zero-padding on $F^{(c)}$ boosts the second-order moment (SOM) of each frequency component at $[u,v]$ as follows, whose strength is measured by averaging over different sampled features.*

$$\begin{aligned} &\forall 0\le u<M,\ 0\le v<N, u+v\neq 0 \\ &\qquad SOM(H^{(c)}_{uv}) - SOM(G^{(c)}_{uv}) = a^2\tau^2_{uv}, \end{aligned} \tag{9}$$

where $SOM(H^{(c)}_{uv}) = \mathbb{E}[H^{(c)}_{uv}\overline{H}^{(c)}_{uv}]$ denotes the second-order moment of $H^{(c)}_{uv}$; $\tau_{uv} = \frac{\sin(\frac{Mu\pi}{M'})}{\sin(\frac{u\pi}{M'})}\frac{\sin(\frac{Nv\pi}{N'})}{\sin(\frac{v\pi}{N'})} \in \mathbb{R}$. Note that for the fundamental frequency $u=v=0$, $SOM(H^{(c)}_{00}) = SOM(G^{(c)}_{00})$.

(Conclusion) According to the rule of the forward propagation in Equation (4) and the change of $T^{(l,uv)}$ in Equation (5), the zero-padding operation boosts the SOM of low-frequency components, because τ^2_{uv} is large for low frequencies. This exhibits the trend of encoding low-frequency components of the input sample.

4.3. Effects of the upsampling operation

Let the l-th intermediate-layer feature map $\mathbf{F} \in \mathbb{R}^{C_l\times M_0\times N_0}$ pass through an upsampling layer to extend its width and height to $M\times N$, subject to $M = M_0\cdot ratio$, $N = N_0\cdot ratio$ as follows.

$$\begin{aligned} &\forall c, m^*, n^*, \\ &\widetilde{F}^{(c)}_{m^*n^*} = \begin{cases} F^{(c)}_{mn}, & \mathrm{mod}(m^*, ratio) = \mathrm{mod}(n^*, ratio) = 0 \\ 0, & otherwise \end{cases} \\ &s.t.\ \ m = \frac{m^*}{ratio};\ n = \frac{n^*}{ratio} \end{aligned} \tag{10}$$

Theorem 4.4. *(Proof in Appendix A.6) Let $\boldsymbol{G} = [G^{(1)}, G^{(2)}, \ldots, G^{(C_l)}] \in \mathbb{C}^{C_l\times M_0\times N_0}$ denote spectrums of the C_l channels of feature $\boldsymbol{F}$. Then, spectrums $\boldsymbol{H} = [H^{(1)}, H^{(2)}, \ldots, H^{(C_l)}] \in \mathbb{C}^{C_l\times M\times N}$ of the output feature $\widetilde{\boldsymbol{F}}$ can be computed as follows.*

$$\begin{aligned} \forall c,u,v,\quad & H^{(c)}_{u+(s-1)M_0, v+(t-1)N_0} = G^{(c)}_{uv} \\ & s.t.\ \ s=1,\ldots,\frac{M}{M_0};\ t=1,\ldots,\frac{N}{N_0} \end{aligned} \tag{11}$$

Theorem 4.4 shows that the upsampling operation repeats the strong magnitude of the fundamental frequency $G^{(c)}_{00}$ of the lower layer to different frequency components $\forall c, H^{(c)}_{u^*v^*}$ of the higher layer, where $u^* = 0, M_0, 2M_0, \ldots; v^* = 0, N_0, 2N_0, \ldots$. Besides Figure 1(b), Appendix C.2 shows such a phenomenon on more datasets.

(Conclusion) The upsampling operation makes the upconvolution operation generate a feature spectrum, in which strong signals of the input periodically appear at certain frequencies. Such strong periodic signals hurt the representation capacity of the network.

More crucially, according to the spectrum propagation in Corollary 3.3, such periodic frequency components can be further propagated to upper layers. Thus, Corollary 3.3 may provide some clues to differentiate real samples and the generated samples.

4.4. Difficulty of representing specific frequencies

Based on the propagation rule in frequency in Section 3, we discover a further counter-intuitive phenomenon, *i.e.*, in the scenario of an auto-encoder, if salient frequency components in the input sample and salient frequency components in the target output for regression have a small shift, then the decoder usually cannot be effectively learned.

Let us consider the input $x \in \mathbb{R}^{M\times N}$ with a single channel to simplify the proof. Then, $G \in \mathbb{C}^{M\times N}$, $H \in \mathbb{C}^{M\times N}$, and $H^* \in \mathbb{C}^{M\times N}$ denote spectrums of the input, the output, and the target image to fit, respectively. In a traditional auto-encoder, people usually set the target image the same as the

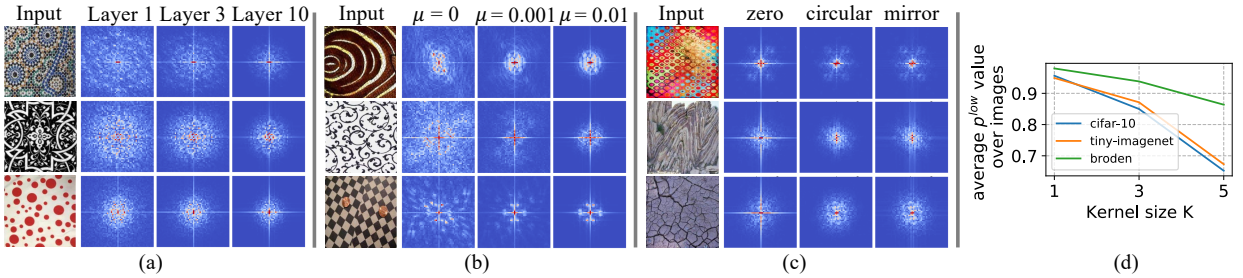


Figure 4. (a) A higher layer of a network usually generated features with more low-frequency components, but with less high-frequency components. (b) A network whose convolutional weights have a mean value significantly biased from 0 usually strengthened low-frequency components, but weakened high-frequency components. (c) A network with zero-padding operations usually strengthened more low-frequency components than a network with circular padding operations. (d) A small kernel size K usually made the network learn a higher proportion p^{low} of low-frequency components. Here, each magnitude map of the feature spectrum was averaged over all channels. For clarity, we moved low frequencies to the center of the spectrum map, and moved high frequencies to corners of the spectrum map. Besides, we only visualized components in the center of the spectrum map with the range of relatively low frequencies[7] in Ω^{low} for clarity.

input image, thereby $H^* = G$. Whereas, let us slightly shift a salient frequency component $[u_1, v_1]$, which is denoted by $G_{u_1v_1} \in \mathbb{C}$, to its neighboring frequency $[u_2, v_2]$ to construct H^* and obtain the target image. Because the frequency component $G_{u_1v_1}$ is salient, we can consider there is a significant increase of $|H^*_{u_2v_2}|$ and a significance decrease of $|H^*_{u_1v_1}|$, compared with the traditional setting of $H^* = G$. Thus, we apply the following setting as a typical case, which simplifies the analysis of the representation problem. That is, $H^*_{u_1v_1} = (1 - A)G^{(u_1v_1)}$, and $H^*_{u_2v_2} = (1 + \overline{A})G^{(u_2v_2)}$, where $A = \alpha e^{i\phi}$, and $\overline{A}$ denotes the conjugate of A; $\alpha > 0$; $\phi < \frac{\pi}{2}$. In this way, we can decrease the significance of $H^*_{u_1v_1}$ and increase the significance of $H^*_{u_2v_2}$.

According to Corollary 3.3, learning an identify function $H = G$ is not difficult for an auto-encoder, because we can just make $\forall u, v, \mathbb{T}^{(uv)(L:1)} = I$. Thus, to investigate the extreme difficulty of learning specific frequencies, let us take the auto-encoder that models the identify function as the baseline network. Then, we further tune the auto-encoder to fit the image with a shifted spectrum H^* and compute the weight changes $\Delta\mathbf{W}$ as the cost of network training. A large weight change[5] $\|\Delta\mathbf{W}\|$ indicates the high difficulty of fitting H^*. Note that we must ensure that parameters $\mathbf{W}$ are real-valued, instead of being complex-valued, when we train the auto-encoder. Theorem 4.5 proves a case that $\mathbf{W}$ is optimized to satisfy $H_{u_1v_1} = H^*_{u_1v_1}$ and $H_{u_2v_2} = H^*_{u_2v_2}$ simultaneously.

Theorem 4.5. *(Proof in Appendix A.7) Let us consider the objective function in the form* $\lambda_1|H_{u_1v_1} - H^*_{u_1v_1}|^2 + \lambda_2|H_{u_2v_2} - H^*_{u_2v_2}|^2$. *We prove specific constrains of* λ_1, λ_2, *and* A *that make the auto-encoder learnable (i.e., ensuring* $\Delta\mathbf{W}$ *is real-valued) and make the objective function can reach zero by a single step of gradient descent, which are shown in Equations (43) and (44) in Appendix A.7. Then, we prove the significance of the weight change* $\Delta\mathbf{W}$, *as follows.*

$$\|\Delta\boldsymbol{W}\| \propto \frac{\alpha MN}{K^2 - \frac{\sin(\frac{K(u_2-u_1)\pi}{M})\sin(\frac{K(v_2-v_1)\pi}{N})}{\sin(\frac{(u_2-u_1)\pi}{M})\sin(\frac{(v_2-v_1)\pi}{N})}} \tag{12}$$

We use the norm of $\Delta\mathbf{W}$ to measure the optimization cost (difficulty) to push the auto-encoder to fit a target image, one of whose frequency component is slightly shifted. Theorem 4.5 shows that the learning difficulty is significantly boosted (*i.e.*, $\|\Delta\mathbf{W}\|$ is much larger) when we shift the target frequency components by a smaller distance $\|[u_1, v_1] - [u_2, v_2]\|$.

5. Experiments

5.1. Verifying the weakening of high frequencies

• **Verifying that a neural network usually learned lowfrequent components first.** Our theorems prove that a cascaded convolutional decoder network weakens the encoding of high-frequency components. In this experiment, we visualized spectrums of the image generated by a decoder network, which showed that the decoder usually learned low-frequency components in early epochs and then shifted its attention to high-frequency components. To this end, we constructed a cascaded convolutional auto-encoder by using the VGG-16 (Simonyan & Zisserman, 2015) as the encoder network. The decoder network contained four upconvolutional layers. Each convolutional/upconvolutional layer in the auto-encoder applied zero-paddings and was followed by a batch normalization layer and an ReLU layer. The auto-encoder was trained on the Tiny-ImageNet dataset (Le & Yang, 2015) using the mean squared error (MSE) loss for image reconstruction[6]. Our theorem was verified by the well-known phenomenon in Figure 1(a), *i.e.*, an auto-encoder

[5] We represent parameters of multiple layers as a vector.

[6] Please see Appendix C.10 for the number of epochs for the training of each model and its fitting error.

usually first generated images with low-frequency components, and then gradually generated more high-frequency components. Results on more datasets in Appendix C.1 yielded similar conclusions.

• **Verifying that the zero-padding operation strengthened the encoding of low-frequency components.** To this end, we compared feature spectrums between the network with zero-padding operations and the network without zero-padding operations. Therefore, we constructed the following three baseline networks. The first baseline network contained 5 convolutional layers, and each layer applied zero-paddings. Each convolutional layer contained 16 convolutional kernels (kernel size was 7×7), except for the last layer containing 3 convolutional kernels. The second and the third baseline networks were constructed by replacing all zero-padding operations with circular padding operations and replacing all zero-padding operations with mirror padding operations, respectively. Results on the Broden (Bau et al., 2017) dataset in Figure 4(c) show that the network with zero-padding operations encoded more significant low-frequency components than the network with circular padding operations. The mirror padding operation also enhanced the significance of low-frequency components, to some extent. Results on more datasets in Appendix C.3 yielded similar conclusions.

• **Verifying factors that strengthened low-frequency components.** Previous studies (Ruderman, 1994) have empirically found that natural images were dominated by low-frequency components. Therefore, according to Corollaries 3.3 and 3.4, we know that if the cascaded convolutional decoder is trained on natural images, then the decoder is more likely to strengthen low-frequencies. Please see Appendix B.4 for more discussions. Besides, we conducted experiments to verify the following three factors that were found to strengthen low frequencies.

(1) Verifying that a deep network strengthened low-frequency components. To this end, we constructed a network with 50 convolutional layers. Each convolutional layer applied zero-paddings to avoid changing the size of feature maps, and was followed by an ReLU layer. We conducted this experiment on three datasets, including CIFAR-10 (Krizhevsky et al., 2009), Tiny-ImageNet, and Broden datasets, respectively. The exponential increase of $\mathbb{T}^{(uv)(L:1)}$ along with the network depth L indicated that the frequency component of the network output also increased exponentially along with L. Therefore, for the frequency component $\mathbf{h}^{(uv)}$ generated by the l-th layer in a real decoder network, we measured its second-order moment $SOM(\mathbf{h}^{(uv)})$. Figure 5 shows that $SOM(\mathbf{h}^{(uv)})$ increased along with the layer number in an exponential manner.

Besides, we visualized feature spectrums of different convolutional layers, which verified the claim that a deep decoder network strengthened the encoding of low-frequency components of the input sample. Results on the Broden dataset in Figure 4(a) show that magnitudes of low frequencies increased along with the network layer number. Results on more datasets in Appendix C.4 yielded similar conclusions.

(2) Verifying that a larger absolute mean value μ_l of each l-th layer's parameters strengthened low-frequency components. To this end, we compared spectrums of output features, when we set convolution parameters with different mean values μ_l. Therefore, we applied the network architecture used in the verification of the zero-padding's effects, but we changed the kernel size to 9×9. Based on this architecture, we constructed three networks, whose parameters were sampled from Gaussian distributions $\mathcal{N}(\mu = 0, \sigma^2 = 0.01^2)$, $\mathcal{N}(\mu = 0.001, \sigma^2 = 0.01^2)$, and $\mathcal{N}(\mu = 0.01, \sigma^2 = 0.01^2)$, respectively. Results on the Broden dataset in Figure 4(b) show that magnitudes of low-frequency components increased along with the absolute mean value of parameters. In addition, Appendix C.5 shows results on more datasets, which also yielded similar conclusions.

We also conducted experiments to measure the effects of large absolute mean values on layers with different depth. Results in Appendix C.8 show that no matter which layer had parameters of a large absolute mean value, there was no significant difference in weakening the encoding of high-frequency components. It was because different convolutional layers, including both shallow and deep layers, theoretically had similar roles in affecting the frequency representation of the entire network, according to Corollary 3.3.

(3) Verifying that a small kernel size K strengthened low-frequency components. To this end, we compared feature spectrums of networks with different kernel sizes. Therefore, we constructed three networks with kernel sizes of 1×1, 3×3, and 5×5. Each network contained 5 convolutional layers, each layer contained 16 convolutional kernels, except for the last layer containing 3 kernels. We used the metric $p^{\text{low}} = \frac{\sum_{[u,v]\in\Omega^{\text{low}}} \mathbb{E}_c[|H_{uv}^{(c)}|^2]}{\sum_{uv} \mathbb{E}_c[|H_{uv}^{(c)}|^2]}$ to measure the ratio of low-frequency components to all frequencies, where Ω^{low} denoted the set of low-frequency components[7]. Figure 4(d) reports the average p^{low} value over all images. Results show that the network with a small kernel size encoded more low-frequency components.

5.2. Verifying the difficulty of fitting shifted frequencies

This experiment was conducted to verify the difficulty of learning an auto-encoder, when salient frequency components of the target output had a small shift from the spec-

[7]Low-frequencies $[u, v] \in \Omega^{\text{low}}$ were included in $u \in \{u|0 \le u < \frac{M}{8}\} \cup \{u|\frac{7M}{8} \le u < M\}$ and $v \in \{v|0 \le v < \frac{N}{8}\} \cup \{v|\frac{7N}{8} \le v < N\}$.

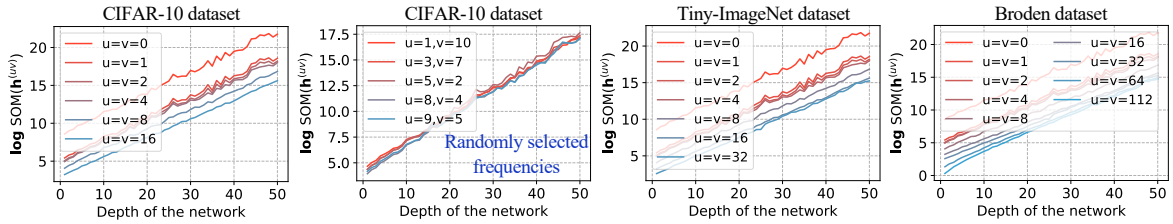


Figure 5. The exponential increase of the SOM of feature spectrums, $SOM(\mathbf{h}^{(uv)})$, along with the network depth, *i.e.*, the linear increase of $\mathbf{log}SOM(\mathbf{h}^{(uv)})$ along with the network depth.

trum of the input. For testing, we constructed an input image $x \sim \mathcal{N}(\mathbf{0}, \Sigma = 0.01 \cdot \mathbf{I})$ following a Gaussian distribution, which made each frequency component of x have weak strength. We then selected six low frequencies within the range of $u \in \{u|0 \le u \le 3\} \cup \{u|60 < u \le 63\}$ and $v \in \{v|0 \le v \le 3\} \cup \{v|60 < v \le 63\}$, and we set significant components for these six low frequencies. Then, we constructed the target image for regression by shifting each salient frequency component $[u, v]$ in x to $[u + \Delta u, v]$ or $[u - \Delta u, v]$ towards higher frequencies[8]. Here, we set $\Delta u = 1, 4, 8$ to generate three target images, respectively. We trained an auto-encoder with five convolutional layers on each pair of the input image and the target image. Each intermediate convolutional layer contained 64 convolutional kernels (with the kernel size 3×3) and followed by an ReLU layer.

We used the metric $\Delta x_{mn} = \left\| [x^{*(1)}_{mn}, x^{*(2)}_{mn}, \ldots, x^{*(C)}_{mn}] - [\hat{x}^{(1)}_{mn}, \hat{x}^{(2)}_{mn}, \ldots, \hat{x}^{(C)}_{mn}] \right\|_2$ to measure the fitting error on the pixel $[m, n]$ between the target output $x^* \in \mathbb{R}^{C \times M \times N}$ and the network output $\hat{x} \in \mathbb{R}^{C \times M \times N}$. Figure 1(c) shows the heatmap of the fitting error Δx_{mn}, and the histogram of fitting errors $|\Delta x_{mn}|$ over different pixels. Results show that when the target frequency components were shifted by a smaller distance, it was more difficult to learn a decoder to fit the target image, *i.e.,* yielding larger fitting errors in Figure 1(c.1,c.2). Besides, Figure 1(c.3) reports the average weight change $\|\Delta \mathbf{W}\|$ over different DNNs, which was decreased along with the shifting distance. This also verified that it took more effort to learn a decoder to fit the target image, when the target frequency components were shifted by a smaller distance.

5.3. Verifying the repeat of certain frequencies

We conducted experiments to verify the problem that the upsampling operation made a decoder network repeat strong signals at certain frequencies of the generated image in Theorem 4.4. To this end, we compared feature spectrums between the input spectrum and the output spectrum of the upsampling layer. We also conducted experiments on the auto-encoder introduced above[6]. Figure 1(b) shows that the decoder network repeated strong signals at certain frequencies of the generated image. Results on more datasets in Appendix C.2 yielded similar conclusions.

[8]Please see Appendix C.9 for details about the frequency shift.

6. Conclusion

In this paper, we have reformulated the rule for the forward propagation of a cascaded convolutional decoder network in the frequency domain. Based on such propagation rules, we have discovered and theoretically proven that both the convolution operation and the zero-padding operation strengthen low-frequency components in the decoder. The upsampling operation repeats the strong magnitude of the fundamental frequency in the input feature to different frequencies of the spectrum of the output feature map. Besides, we also discover and prove the difficulty of pushing an auto-encoder to fit specific frequency components in the target output, which have a small shift from the spectrum of the input. Such properties may hurt the representation capacity of a convolutional decoder network. Experiments on ReLU networks have verified our theoretical proofs. Note that our findings can explain general trends of networks with above three operations, but cannot derive a deterministic property of a specific network, and cannot be extended to networks for image classification, because we have not derived the property of the max-pooling operation.

Acknowledgements. This work is partially supported by the National Nature Science Foundation of China (62276165, 62206170), National Key R&D Program of China (2021ZD0111602), Shanghai Natural Science Foundation (21JC1403800,21ZR1434600), National Nature Science Foundation of China (U19B2043), and the Alibaba Group through Alibaba Innovative Research Program.

References

Amjad, R. A. and Geiger, B. C. Learning representations for neural network-based classification using the information bottleneck principle. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2225–2239, 2019.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of

deep visual representations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3319–3327, 2017.

Deng, H., Ren, Q., Chen, X., Zhang, H., Ren, J., and Zhang, Q. Discovering and explaining the representation bottleneck of dnns. In *International Conference on Learning Representations*, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Jain, A. K. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

Kreutz-Delgado, K. The complex gradient operator and the cr-calculus. *arXiv preprint arXiv:0906.4835*, 2009.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Li, X., Chen, S., and Yang, J. Understanding the disharmony between weight normalization family and weight decay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Lin, Y., Jiang, H., et al. Bandlimiting neural networks against adversarial attacks. *arXiv preprint arXiv:1905.12797*, 2019.

Luo, T., Ma, Z., Xu, Z.-Q. J., and Zhang, Y. Theory of the frequency principle for general deep neural networks. *arXiv preprint arXiv:1906.09235*, 2019.

Ma, C., Wu, L., et al. Machine learning from a continuous viewpoint, i. *Science China Mathematics*, 63(11):2233–2266, 2020.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2019.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Reinhard, H. and Fatih, F. Y. Early stopping in deep networks: Double descent and how to eliminate it. In *International Conference on Learning Representations*, 2020.

Ruderman, D. L. The statistics of natural images. *Network: computation in neural systems*, 5(4):517, 1994.

Shwartz-Ziv, R. and Tishby, N. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Sundararajan, D. *The discrete Fourier transform: theory, algorithms and applications*. World Scientific, 2001.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. IEEE, 2015.

Tse, D. and Viswanath, P. *Fundamentals of wireless communication*. Cambridge university press, 2005.

Van Laarhoven, T. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

Wang, H., Wu, X., Huang, Z., and Xing, E. P. High-frequency component helps explain the generalization of convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8684–8694, 2020.

Wolchover, N. and Reading, L. New theory cracks open the black box of deep learning. *Quanta Magazine*, 3, 2017.

Xu, Z.-Q. J., Zhang, Y., Luo, T., Xiao, Y., and Ma, Z. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019a.

Xu, Z.-Q. J., Zhang, Y., and Xiao, Y. Training behavior of deep neural network in frequency domain. In *International Conference on Neural Information Processing*, pp. 264–274. Springer, 2019b.

Yin, D., Gontijo Lopes, R., Shlens, J., Cubuk, E. D., and Gilmer, J. A fourier perspective on model robustness in computer vision. *Advances in Neural Information Processing Systems*, 32, 2019.

A. Proofs of our theoretical findings

We first introduce an important equation, which is widely used in the following proofs.

Lemma A.1. Given N complex numbers, $e^{in\theta}$, $n = 0, 1, \ldots, N-1$, the sum of these N complex numbers is given as follows.

$$\forall \theta \in \mathbb{R}, \qquad \sum_{n=0}^{N-1} e^{in\theta} = \frac{\sin(\frac{N\theta}{2})}{\sin(\frac{\theta}{2})} e^{i\frac{(N-1)\theta}{2}} \tag{13}$$

Specifically, when $N\theta = 2k\pi, k \in \mathbb{Z}, -N < k < N$, we have

$$\begin{aligned} \forall \theta \in \mathbb{R}, \quad \sum_{n=0}^{N-1} e^{in\theta} = \frac{\sin(\frac{N\theta}{2})}{\sin(\frac{\theta}{2})} e^{i\frac{(N-1)\theta}{2}} = N\delta_\theta; \quad \text{s.t. } N\theta = 2k\pi, k \in \mathbb{Z}, -N < k < N, \\ \textit{where} \quad \delta_\theta = \begin{cases} 1, & \theta = 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{14}$$

We prove Lemma A.1 as follows.

Proof. First, let us use the letter $S \in \mathbb{C}$ to denote the term of $\sum_{n=0}^{N-1} e^{in\theta}$.

$$S = \sum_{n=0}^{N-1} e^{in\theta}$$

Therefore, $e^{i\theta}S$ is formulated as follows.

$$e^{i\theta}S = \sum_{n=1}^{N} e^{in\theta} \in \mathbb{C}$$

Then, S can be computed as $S = \frac{e^{i\theta}S - S}{e^{i\theta} - 1}$. Therefore, we have

$$\begin{aligned} S &= \frac{e^{i\theta}S - S}{e^{i\theta} - 1} \\ &= \frac{\sum_{n=1}^{N} e^{in\theta} - \sum_{n=0}^{N-1} e^{in\theta}}{e^{i\theta} - 1} \\ &= \frac{e^{iN\theta} - 1}{e^{i\theta} - 1} \\ &= \frac{e^{i\frac{N\theta}{2}} - e^{-i\frac{N\theta}{2}}}{e^{i\frac{\theta}{2}} - e^{-i\frac{\theta}{2}}} e^{i\frac{(N-1)\theta}{2}} \\ &= \frac{(e^{i\frac{N\theta}{2}} - e^{-i\frac{N\theta}{2}})/2i}{(e^{i\frac{\theta}{2}} - e^{-i\frac{\theta}{2}})/2i} e^{i\frac{(N-1)\theta}{2}} \\ &= \frac{\sin(\frac{N\theta}{2})}{\sin(\frac{\theta}{2})} e^{i\frac{(N-1)\theta}{2}} \end{aligned}$$

Therefore, we prove that $\sum_{n=0}^{N-1} e^{in\theta} = \frac{\sin(\frac{N\theta}{2})}{\sin(\frac{\theta}{2})} e^{i\frac{(N-1)\theta}{2}}$.

Then, we prove the special case that when $N\theta = 2k\pi, k \in \mathbb{Z}, -N < k < N$, $\sum_{n=0}^{N-1} e^{in\theta} = N\delta_\theta = \begin{cases} N, & \theta = 0 \\ 0, & \text{otherwise} \end{cases}$, as follows.

When $\theta = 0$, we have

$$\begin{aligned}\lim_{\theta\to 0}\sum_{n=0}^{N-1} e^{in\theta} &= \lim_{\theta\to 0}\frac{\sin(\frac{N\theta}{2})}{\sin(\frac{\theta}{2})}e^{i\frac{(N-1)\theta}{2}}\\ &= \lim_{\theta\to 0}\frac{\sin(\frac{N\theta}{2})}{\sin(\frac{\theta}{2})}\\ &= N\end{aligned}$$

When $\theta \neq 0$, and $N\theta = 2k\pi, k \in \mathbb{Z}, -N < k < N$, we have

$$\begin{aligned}\sum_{n=0}^{N-1} e^{in\theta} &= \frac{\sin(\frac{N\theta}{2})}{\sin(\frac{\theta}{2})}e^{i\frac{(N-1)\theta}{2}}\\ &= \frac{\sin(k\pi)}{\sin(\frac{k\pi}{N})}e^{i\frac{(N-1)k\pi}{N}}\\ &= 0\end{aligned}$$

□

In the following proofs, the following two equations are widely used, which are derived based on Lemma A.1.

$$\begin{aligned}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi} &= \sum_{m=0}^{M-1} e^{im(-\frac{u2\pi}{M})}\sum_{n=0}^{N-1} e^{in(-\frac{v2\pi}{N})}\\ &= (M\delta_{-\frac{u2\pi}{M}})(N\delta_{-\frac{v2\pi}{N}}) \quad \text{//According to Equation (14)}\\ &= \begin{cases} MN, & u = v = 0\\ 0, & \text{otherwise}\end{cases}\end{aligned}$$

To simplify the representation, **let δ_{uv} be the simplification of $\delta_{-\frac{u2\pi}{M}}\delta_{-\frac{v2\pi}{N}}$ in the following proofs.** Therefore, we have

$$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi} = MN\delta_{uv} = \begin{cases} MN, & u = v = 0\\ 0, & \text{otherwise}\end{cases} \tag{15}$$

Similarly, we derive the second equation as follows.

$$\begin{aligned}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{i(\frac{(u-u')m}{M}+\frac{(v-v')n}{N})2\pi} &= \sum_{m=0}^{M-1} e^{im(\frac{(u-u')2\pi}{M})}\sum_{n=0}^{N-1} e^{in(\frac{(v-v')2\pi}{N})}\\ &= MN\delta_{\frac{(u-u')2\pi}{M}}\delta_{\frac{(v-v')2\pi}{N}} \quad \text{//According to Equation (14)}\\ &= MN\delta_{u-u'}\delta_{v-v'}\\ &= \begin{cases} MN, & u' = u; v' = v\\ 0, & \text{otherwise}\end{cases}\end{aligned} \tag{16}$$

A.1. Proof of Theorem 3.2

In this section, we prove Theorem 3.2 in Section 3 of the main paper, as follows.

Proof. Given each c-th channel of the feature spectrum $G^{(c)}$, the corresponding feature $F^{(c)}$ in the time domain can be computed as follows.

$$F^{(c)}_{mn} = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} G^{(c)}_{uv} e^{i(\frac{um}{M}+\frac{vn}{N})2\pi}$$

Then, let us conduct the convlution operation (in Equation (1) in the main paper) on feature $\mathbf{F} = [F^{(1)}, F^{(2)}, \ldots, F^{(C)}]$, in order to obtain the output feature $\widetilde{\mathbf{F}} \in \mathbb{R}^{D \times M' \times N'}$.

$$
\begin{aligned}
&\forall d = 1, 2, \ldots, D; 0 \leq m < M'; 0 \leq n < N';\\
&\quad \tilde{F}_{mn}^{(d)} = b^{(d)} + \sum_{c=1}^{C} \sum_{t=0}^{K-1} \sum_{s=0}^{K-1} W_{cts}^{ker=d} F_{m+t,n+s}^{(c)}\\
&\quad\quad = b^{(d)} + \sum_{c=1}^{C} \sum_{t=0}^{K-1} \sum_{s=0}^{K-1} W_{cts}^{ker=d} \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} G_{uv}^{(c)} e^{i(\frac{u(m+t)}{M} + \frac{v(n+s)}{N})2\pi}\\
&\quad\quad = b^{(d)} + \sum_{c=1}^{C} \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} G_{uv}^{(c)} e^{i(\frac{um}{M} + \frac{vn}{N})2\pi} \sum_{t=0}^{K-1} \sum_{s=0}^{K-1} W_{cts}^{ker=d} e^{i(\frac{ut}{M} + \frac{vs}{N})2\pi}\\
&\quad\quad = b^{(d)} + \sum_{c=1}^{C} \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T_{dc}^{(uv)} G_{uv}^{(c)} e^{i(\frac{um}{M} + \frac{vn}{N})2\pi}
\end{aligned}
$$

Then, let us conduct the DFT on each channel of $\widetilde{\mathbf{F}}$, in order to obtain feature spectrums $H_{u'v'}^{(d)}$ of $\widetilde{\mathbf{F}}$.

$$
\begin{aligned}
&\forall d = 1, 2, \ldots, D;\ 0 \leq u' < M';\ 0 \leq v' < N';\\
&\quad H_{u'v'}^{(d)} = \sum_{m=0}^{M'-1} \sum_{n=0}^{N'-1} \widetilde{F}_{mn}^{(l,d)} e^{-i(\frac{u'm}{M'} + \frac{v'n}{N'})2\pi}\\
&\quad\quad = \sum_{m=0}^{M'-1} \sum_{n=0}^{N'-1} e^{-i(\frac{u'm}{M'} + \frac{v'n}{N'})2\pi} (b^{(d)} + \sum_{c=1}^{C} \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T_{dc}^{(uv)} G_{uv}^{(c)} e^{i(\frac{um}{M} + \frac{vn}{N})2\pi}) \quad \text{//Equation (15)}\\
&\quad\quad = M'N'b^{(d)}\delta_{u'v'} + \sum_{c=1}^{C} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T_{dc}^{(uv)} G_{uv}^{(c)} \frac{1}{MN} \sum_{m=0}^{M'-1} \sum_{n=0}^{N'-1} e^{i((\frac{u}{M} - \frac{u'}{M'})m + (\frac{v}{N} - \frac{v'}{N'})n)2\pi}\\
&\quad \text{// Let } \alpha_{u'v'uv} = \frac{1}{MN} \sum_{m=0}^{M'-1} \sum_{n=0}^{N'-1} e^{i((\frac{u}{M} - \frac{u'}{M'})m + (\frac{v}{N} - \frac{v'}{N'})n)2\pi}\\
&\quad\quad = M'N'b^{(d)}\delta_{u'v'} + \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \alpha_{u'v'uv} \sum_{c=1}^{C} T_{dc}^{(uv)} G_{uv}^{(c)}
\end{aligned}
$$

When the convlution operation does not apply paddings, and its stride size is 1, $M' = M - K + 1, N' = N - K + 1$. In this way, $\alpha_{u'v'uv}$ can be rewritten as follows.

$$
\begin{aligned}
\alpha_{u'v'uv} &= \frac{1}{MN} \sum_{m=0}^{M'-1} \sum_{n=0}^{N'-1} e^{i((\frac{u}{M} - \frac{u'}{M'})m + (\frac{v}{N} - \frac{v'}{N'})n)2\pi}\\
&//M' = M - K + 1, N' = N - K + 1\\
&= \frac{1}{MN} \sum_{m=0}^{M-K} \sum_{n=0}^{N-K} e^{i((\frac{u}{M} - \frac{u'}{M-K+1})m + (\frac{v}{N} - \frac{v'}{N-K+1})n)2\pi}\\
&= \frac{1}{MN} \sum_{m=0}^{M-K} e^{i(\frac{u}{M} - \frac{u'}{M-K+1})2\pi m} \sum_{n=0}^{N-K} e^{i(\frac{v}{N} - \frac{v'}{N-K+1})2\pi n}\\
&//\text{According to Equation (13)}\\
&= \frac{1}{MN} \frac{\sin((M-K)\lambda_{uu'}\pi)}{\sin(\lambda_{uu'}\pi)} \frac{\sin((N-K)\gamma_{vv'}\pi)}{\sin(\gamma_{vv'}\pi)} e^{i((M-K)\lambda_{uu'} + (N-K)\gamma_{vv'})\pi}
\end{aligned}
\tag{17}
$$

where $\lambda_{uu'} = \frac{(u-u')M - u(K-1)}{M(M-K+1)}$, $\gamma_{vv'} = \frac{(v-v')N - v(K-1)}{N(N-K+1)}$.

Therefore, we prove that the vector $\mathbf{h}^{(u'v')} = [H_{u'v'}^{(1)}, H_{u'v'}^{(2)}, \ldots, H_{u'v'}^{(D)}]^\top \in \mathbb{C}^D$ can be computed as follows.

$$
\forall d = 1, 2, \ldots, D; \quad \mathbf{h}^{(u'v')} = \delta_{u'v'} M'N' \mathbf{b} + \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \alpha_{u'v'uv} T^{(uv)} \mathbf{g}^{(uv)}
$$

Furthermore, based on Assumption 3.1, the convolution operation does not change the size of the feature map, *i.e.*, $M' = M$, $N' = N$. In this case, $\alpha_{u'v'uv}$ can be computed as follows.

$$\begin{aligned}
\alpha_{u'v'uv} &= \frac{1}{MN}\sum_{m=0}^{M'-1}\sum_{n=0}^{N'-1} e^{i((\frac{u}{M}-\frac{u'}{M'})m+(\frac{v}{N}-\frac{v'}{N'})n)2\pi} \\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{i(\frac{(u-u')m}{M}+\frac{(v-v')n}{N})2\pi} \quad //M'=M, N'=N \\
&= \frac{1}{MN}\sum_{m=0}^{M-1} e^{i(\frac{(u-u')2\pi}{M})m}\sum_{n=0}^{N-1} e^{i(\frac{(v-v')2\pi}{N})n} \quad //\text{According to Equation (16)} \\
&= \delta_{u-u'}\delta_{v-v'}
\end{aligned} \tag{18}$$

where $\delta_{u-u'} = \begin{cases} 1, u'=u \\ 0, \text{otherwise} \end{cases}$; $\delta_{v-v'} = \begin{cases} 1, v'=v \\ 0, \text{otherwise} \end{cases}$.

Therefore, $\mathbf{h}^{(u'v')}$ can be computed as follows.

$$\begin{aligned}
\mathbf{h}^{(u'v')} &= \sum_{u=0}^{M'-1}\sum_{v=0}^{N'-1} \alpha_{u'v'uv} T^{(u'v')}\mathbf{g}^{(u'v')} + \delta_{u'v'}M'N'\mathbf{b} \\
&= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \delta_{u-u'}\delta_{v-v'} T^{(u'v')}\mathbf{g}^{(u'v')} + \delta_{u'v'}MN\mathbf{b} \\
&= T^{(u'v')}\mathbf{g}^{(u'v')} + MN\mathbf{b}\delta_{u'v'}
\end{aligned}$$

Then, we prove that $\mathbf{h}^{(uv)} = T^{(uv)}\mathbf{g}^{(uv)} + MN\mathbf{b}\delta_{uv}$. □

A.2. Proof of Corollary 3.3

In this section, we prove Corollary 3.3 in Section 3 of the main paper, as follows.

Proof. Let $\mathbf{G}^{(l)} = [G^{(l,1)}, G^{(l,2)}, \cdots, G^{(l,C_l)}] \in \mathbb{C}^{C_l \times M \times N}$ denote feature spectrums of the l-th layer. Let $\mathbf{g}^{(l,uv)} = [G_{uv}^{(l,1)}, G_{uv}^{(l,2)}, \cdots, G_{uv}^{(l,C_l)}]^\top \in \mathbb{C}^{C_l}$ denote the frequency component at the frequency $[u, v]$. When $l = 0$, $\mathbf{g}^{(0,uv)}$ denotes the frequency component of the input sample. When $l = L$, $\mathbf{g}^{(L,uv)}$ denotes the frequency component of the network output.

Based on Theorem 3.2, $\mathbf{g}^{(l,uv)}$ can be computed as follows.

$$\forall l = 1, 2, \ldots, L, \quad \mathbf{g}^{(l,uv)} = T^{(l,uv)}\mathbf{g}^{(l-1,uv)} + \delta_{uv}MN\mathbf{b}^{(l)}$$

Then, the frequency component $\mathbf{g}^{(L,uv)}$ of the network output can be computed as follows.

$$\begin{aligned}
\mathbf{g}^{(L,uv)} &= T^{(L,uv)}\mathbf{g}^{(L-1,uv)} + \delta_{uv}MN\mathbf{b}^{(L)} \\
&= T^{(L,uv)}(T^{(L-1,uv)}\mathbf{g}^{(L-2,uv)} + \delta_{uv}MN\mathbf{b}^{(L-1)}) + \delta_{uv}MN\mathbf{b}^{(L)} \\
&= T^{(L,uv)}T^{(L-1,uv)}\mathbf{g}^{(L-2,uv)} + T^{(L,uv)}\delta_{uv}MN\mathbf{b}^{(L-1)} + \delta_{uv}MN\mathbf{b}^{(L)} \\
&= \cdots \\
&= T_{dc}^{(l,uv)}\cdots T^{(1,uv)}\mathbf{g}^{(0,uv)} + MNT_{dc}^{(l,uv)}\cdots T^{(2,uv)}\mathbf{b}^{(1)}\delta_{uv} + \cdots + MN\mathbf{b}^{(L)}\delta_{uv} \\
&= T_{dc}^{(l,uv)}\cdots T^{(1,uv)}\mathbf{g}^{(0,uv)} + \delta_{uv}MN(T_{dc}^{(l,uv)}\cdots T^{(2,uv)}\mathbf{b}^{(1)} + \cdots + MN\mathbf{b}^{(L)})
\end{aligned}$$

Let $\mathbb{T}^{(uv)(L:1)} = T_{dc}^{(l,uv)}\cdots T^{(2,uv)}T^{(1,uv)}$ and $\boldsymbol{\beta} = MN\big(\mathbf{b}^{(L)} + \sum_{j=2}^{L}\mathbb{T}^{(00)(L:j)}\mathbf{b}^{(j-1)}\big)$. Let $\mathbf{h}^{(uv)} = \mathbf{g}^{(L,uv)}$ denote the frequency component of the network output, and let $\mathbf{g}^{(uv)} = \mathbf{g}^{(0,uv)}$ denote the frequency component of the input sample. Then, we prove that $\mathbf{h}^{(uv)}$ can be computed as follows.

$$\mathbf{h}^{(uv)} = \mathbb{T}^{(uv)(L:1)}\mathbf{g}^{(uv)} + \delta_{uv}\boldsymbol{\beta}$$

□

A.3. Proof of Corollary 3.4

In this section, we prove Corollary 3.4 in Section 3 of the main paper, as follows.

Proof. **First, we focus on a single convolutional layer.**

According to the DFT and the inverse DFT, we can obtain the mathematical relationship between $G_{uv}^{(l,c)}$ and $F_{mn}^{(l,c)}$, and the mathematical relationship between $T_{dc}^{(l,uv)}$ and $W_{cts}^{(l)[\text{ker}=d]}$, as follows.

$$\begin{cases} G_{uv}^{(l,c)} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F_{mn}^{(l,c)} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi} \\ F_{mn}^{(l,c)} = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} G_{uv}^{(l,c)} e^{i(\frac{um}{M}+\frac{vn}{N})2\pi} \end{cases} \quad \begin{cases} T_{dc}^{(l,uv)} = \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W_{cts}^{(l)[\text{ker}=d]} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\ W_{cts}^{(l)[\text{ker}=d]} = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T_{dc}^{(l,uv)} e^{-i(\frac{ut}{M}+\frac{vs}{N})2\pi} \end{cases} \tag{19}$$

Based on Equation (19) and the derivation rule for complex numbers (Kreutz-Delgado, 2009), we can obtain the mathematical relationship between $\frac{\partial Loss}{\partial \overline{G}_{uv}^{(l,c)}}$ and $\frac{\partial Loss}{\partial \overline{F}_{mn}^{(l,c)}}$, and the mathematical relationship between $\frac{\partial Loss}{\partial \overline{T}_{dc}^{(l,uv)}}$ and $\frac{\partial Loss}{\partial \overline{W}_{cts}^{(l)[\text{ker}=d]}}$, as follows. Note that when we use gradient descent to optimize a real-valued loss function *Loss* with complex variables, people usually treat the real and imaginary values, $a \in \mathbb{C}$ and $b \in \mathbb{C}$, of a complex variable ($z = a + bi$) as two separate real-valued variables, and separately update these two real-valued variables. In this way, the exact optimization step of z computed based on such a technology is equivalent to $\frac{\partial Loss}{\partial \overline{z}}$. Since $F_{mn}^{(l,c)}$ and $W_{cts}^{(l)[\text{ker}=d]}$ are real numbers, $\frac{\partial Loss}{\partial \overline{F}_{mn}^{(l,c)}} = \frac{\partial Loss}{\partial F_{mn}^{(l,c)}}$ and $\frac{\partial Loss}{\partial \overline{W}_{cts}^{(l)[\text{ker}=d]}} = \frac{\partial Loss}{\partial W_{cts}^{(l)[\text{ker}=d]}}$.

$$\begin{cases} \frac{\partial Loss}{\partial \overline{G}_{uv}^{(l,c)}} = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \frac{\partial Loss}{\partial \overline{F}_{mn}^{(l,c)}} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi} \\ \frac{\partial Loss}{\partial \overline{F}_{mn}^{(l,c)}} = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \frac{\partial Loss}{\partial \overline{G}_{uv}^{(l,c)}} e^{i(\frac{um}{M}+\frac{vn}{N})2\pi} \end{cases} \quad \begin{cases} \frac{\partial Loss}{\partial \overline{T}_{dc}^{(l,uv)}} = \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1} \frac{\partial Loss}{\partial \overline{W}_{cts}^{(l)[\text{ker}=d]}} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\ \frac{\partial Loss}{\partial \overline{W}_{cts}^{(l)[\text{ker}=d]}} = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \frac{\partial Loss}{\partial \overline{T}_{dc}^{(l,uv)}} e^{-i(\frac{ut}{M}+\frac{vs}{N})2\pi} \end{cases} \tag{20}$$

Let us conduct the convolution operation (based on Assumption 3.1) on the feature map $\mathbf{F}^{(l-1)} = [F^{(l-1,1)}, F^{(l-1,2)}, \ldots, F^{(l-1,C)}] \in \mathbb{R}^{C\times M\times N}$, and obtain the output feature map $\mathbf{F}^{(l)} = [F^{(l,1)}, F^{(l,2)}, \ldots, F^{(l,D)}] \in \mathbb{R}^{D\times M\times N}$ of the l-th layer as follows.

$$F_{mn}^{(l,d)} = b^{(d)} + \sum_{c=1}^{C}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W_{cts}^{(l)[\text{ker}=d]} F_{m+t,n+s}^{(l-1,c)} \tag{21}$$

Based on Equation (19) and Equation (20), and the derivation rule for complex numbers (Kreutz-Delgado, 2009), the exact

optimization step of $T_{dc}^{(l,uv)}$ in real implementations can be computed as follows.

$$
\begin{aligned}
&\frac{\partial Loss}{\partial \overline{T}_{dc}^{(l,uv)}} \\
&= \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\frac{\partial Loss}{\partial \overline{W}_{cts}^{(l)[\text{ker}=d]}}e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \quad \text{//Equation (20)} \\
&= \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\frac{\partial Loss}{\partial \overline{F}_{mn}^{(l,d)}}\cdot \overline{F}_{m+t,n+s}^{(l-1,c)}\right)e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \quad \text{//Equation (21)} \\
&\ \text{//Equation (19)} \\
&= \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\frac{\partial Loss}{\partial \overline{F}_{mn}^{(l,d)}}\cdot \frac{1}{MN}\sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\overline{G}_{u'v'}^{(l-1,c)}e^{-i(\frac{u'(m+t)}{M}+\frac{v'(n+s)}{N})2\pi}\right)e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\
&= \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\left(\sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\overline{G}_{u'v'}^{(l-1,c)}e^{-i(\frac{u't}{M}+\frac{v's}{N})2\pi}\cdot \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\frac{\partial Loss}{\partial \overline{F}_{mn}^{(l,d)}}e^{-i(\frac{u'm}{M}+\frac{v'n}{N})2\pi}\right)e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\
&= \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\left(\sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\overline{G}_{u'v'}^{(l-1,c)}\frac{\partial Loss}{\partial \overline{G}_{u'v'}^{(l,d)}}e^{-i(\frac{u't}{M}+\frac{v's}{N})2\pi}\right)e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \quad \text{//Equation (20)} \\
&= \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\overline{G}_{u'v'}^{(l-1,c)}\frac{\partial Loss}{\partial \overline{G}_{u'v'}^{(l,d)}}e^{i(\frac{(u-u')t}{M}+\frac{(v-v')s}{N})2\pi} \\
&= \sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\overline{G}_{u'v'}^{(l-1,c)}\frac{\partial Loss}{\partial \overline{G}_{u'v'}^{(l,d)}}\cdot\frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}e^{i(\frac{(u-u')t}{M}+\frac{(v-v')s}{N})2\pi} \\
&\text{// Let } \chi_{u'v'uv} = \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}e^{i(\frac{(u-u')t}{M}+\frac{(v-v')s}{N})2\pi} \\
&= \sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\chi_{u'v'uv}\overline{G}_{u'v'}^{(l-1,c)}\frac{\partial Loss}{\partial \overline{G}_{u'v'}^{(l,d)}}
\end{aligned}
$$

where $\chi_{u'v'uv}$ can be rewritten as follows.

$$
\begin{aligned}
\chi_{u'v'uv} &= \frac{1}{MN}\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}e^{i(\frac{(u-u')t}{M}+\frac{(v-v')s}{N})2\pi} \\
&= \frac{1}{MN}\sum_{t=0}^{K-1}e^{i\frac{(u-u')2\pi}{M}t}\sum_{s=0}^{K-1}e^{i\frac{(v-v')2\pi}{N}s} \\
&= \frac{1}{MN}\frac{\sin(\frac{K(u-u')\pi}{M})}{\sin(\frac{(u-u')\pi}{M})}\frac{\sin(\frac{K(v-v')\pi}{N})}{\sin(\frac{(v-v')\pi}{N})}\cdot e^{i(\frac{(K-1)(u-u')}{M}+\frac{(K-1)(v-v')}{N})\pi} \quad \text{//According to Equation (13)}
\end{aligned}
$$

Similarly, we computed the gradient of the loss function *w.r.t.* the spectrum map $\overline{G}^{(l-1,c)}$ as follows.

$$
\begin{aligned}
&\frac{\partial Loss}{\partial \overline{G}^{(l-1,c)}_{u'v'}} \\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\frac{\partial Loss}{\partial \overline{F}^{(l-1,c)}_{mn}} e^{-i(\frac{u'm}{M}+\frac{v'n}{N})2\pi} \quad \text{//Equation (20)} \\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\overline{W}^{(l)[\text{ker}=d]}_{cts}\cdot\frac{\partial Loss}{\partial \overline{F}^{(l,d)}_{m-t,n-s}}\right) e^{-i(\frac{u'm}{M}+\frac{v'n}{N})2\pi} \quad \text{//Equation (21)} \\
&\text{//According to Equation (20)} \\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\overline{W}^{(l)[\text{ker}=d]}_{cts}\cdot\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\frac{\partial Loss}{\partial \overline{G}^{(l,d)}_{uv}} e^{i(\frac{u(m-t)}{M}+\frac{v(n-s)}{N})2\pi}\right) e^{-i(\frac{u'm}{M}+\frac{v'n}{N})2\pi} \\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\frac{\partial Loss}{\partial \overline{G}^{(l,d)}_{uv}} e^{i(\frac{um}{M}+\frac{vn}{N})2\pi}\cdot\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\overline{W}^{(l)[\text{ker}=d]}_{cts} e^{-i(\frac{ut}{M}+\frac{vs}{N})2\pi}\right) e^{-i(\frac{u'm}{M}+\frac{v'n}{N})2\pi} \\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\frac{\partial Loss}{\partial \overline{G}^{(l,d)}_{uv}}\overline{T}^{(l,uv)}_{dc} e^{i(\frac{um}{M}+\frac{vn}{N})2\pi}\right) e^{-i(\frac{u'm}{M}+\frac{v'n}{N})2\pi} \quad \text{//Equation (19)} \\
&= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\frac{\partial Loss}{\partial \overline{G}^{(l,d)}_{uv}}\overline{T}^{(l,uv)}_{dc}\cdot\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{i(\frac{(u-u')m}{M}+\frac{(v-v')n}{N})2\pi} \\
&= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\frac{\partial Loss}{\partial \overline{G}^{(l,d)}_{uv}}\overline{T}^{(l,uv)}_{dc}\cdot\delta_{u-u'}\delta_{v-v'} \quad \text{//Equation (16)} \\
&= \frac{\partial Loss}{\partial \overline{G}^{(l,d)}_{u'v'}}\overline{T}^{(l,u'v')}_{dc}
\end{aligned}
$$

Based on the derived $\frac{\partial Loss}{\partial \overline{T}^{(l,uv)}_{dc}} \in \mathbb{C}$ and $\frac{\partial Loss}{\partial \overline{G}^{(l-1,c)}_{u'v'}} \in \mathbb{C}$, we can further compute gradients $\frac{\partial Loss}{\partial (\overline{T}^{(l,uv)})^\top} \in \mathbb{C}^{D\times C}$ and $\frac{\partial Loss}{\partial (\overline{\mathbf{g}}^{(l-1,u'v')})^\top} \in \mathbb{C}^{C}$ as follows.

$$
\frac{\partial Loss}{\partial (\overline{T}^{(l,uv)})^\top} = \sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\chi_{u'v'uv}\overline{\mathbf{g}}^{(l-1,u'v')}\frac{\partial Loss}{\partial (\overline{\mathbf{g}}^{(l,u'v')})^\top} \tag{22}
$$

$$
\frac{\partial Loss}{\partial (\overline{\mathbf{g}}^{(l-1,u'v')})^\top} = \frac{\partial Loss}{\partial (\overline{\mathbf{g}}^{(l,u'v')})^\top}\overline{T}^{(l,u'v')} \tag{23}
$$

Furthermore, **we extend the above proof of a single convolutional layer to a network with L cascaded convolutional layers.** Let $\mathbf{g}^{(l,u'v')}$ denote the frequency component at the frequency $[u', v']$ of the l-th layer's output feature, and let $T^{(l,uv)}$ the matrix computed by the l-th layer's convolutional weights. Then, according to Equation (23), the gradient *w.r.t.* $\overline{\mathbf{g}}^{(l,u'v')}$ can be computed as follows.

$$
\begin{aligned}
\frac{\partial Loss}{\partial (\overline{\mathbf{g}}^{(l,u'v')})^T} &= \frac{\partial Loss}{\partial (\overline{\mathbf{g}}^{(L,u'v')})^T}\overline{T}^{(L,u'v')}\cdots\overline{T}^{(l+1,u'v')} \\
&= \frac{\partial Loss}{\partial (\overline{\mathbf{g}}^{(L,u'v')})^T}\overline{\mathbb{T}}^{(u'v')(L:l+1)}
\end{aligned} \tag{24}
$$

According to Equation (22), the gradient *w.r.t.* $\overline{T}^{(l,uv)}$ can be computed as follows.

$$\begin{aligned}
\frac{\partial Loss}{\partial(\overline{T}^{(l,uv)})^\top} &= \sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\chi_{u'v'uv}\overline{\mathbf{g}}^{(l-1,u'v')}\frac{\partial Loss}{\partial(\overline{\mathbf{g}}^{(l,u'v')})^\top} \\
&= \sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\chi_{u'v'uv}(\overline{\mathbb{T}}^{(u'v')(l-1:1)}\overline{\mathbf{g}}^{(0,u'v')} + \overline{\boldsymbol{\beta}}'\delta_{u'v'})\frac{\partial Loss}{\partial(\overline{\mathbf{g}}^{(L,u'v')})^\top}\overline{\mathbb{T}}^{(u'v')(L:l+1)} \text{//Corollary 3.3} \\
&\quad \text{// Let } \mathbf{g}^{(uv)} = \mathbf{g}^{(0,uv)}; \mathbf{h}^{(uv)} = \mathbf{g}^{(L,uv)} \\
&= \sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\chi_{u'v'uv}(\overline{\mathbb{T}}^{(u'v')(l-1:1)}\overline{\mathbf{g}}^{(u'v')} + \overline{\boldsymbol{\beta}}'\delta_{u'v'})\frac{\partial Loss}{\partial(\overline{\mathbf{h}}^{(u'v')})^\top}\overline{\mathbb{T}}^{(u'v')(L:l+1)}
\end{aligned} \tag{25}$$

Let us use the gradient descent algorithm to update the convlutional weight $W_c^{(l)[\text{ker}=d]}|_n$ of the n-th epoch, the updated frequency spectrum $W_c^{(l)[\text{ker}=d]}|_{n+1}$ can be computed as follows.

$$\forall t,s, \quad W_{cts}^{(l)[\text{ker}=d]}|_{n+1} = W_{cts}^{(l)[\text{ker}=d]}|_n - \eta\cdot\frac{\partial Loss}{\partial\overline{W}_{cts}^{(l)[\text{ker=d}]}}$$

where η is the learning rate. Then, the updated frequency spectrum $T^{(l,uv)}|_{n+1}$ computed based on Equation (20) is given as follows.

$$\begin{aligned}
\Delta T_{dc}^{(l,uv)} &= T_{dc}^{(l,uv)}|_{n+1} - T_{dc}^{(l,uv)}|_n \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W_{cts}^{(l)[\text{ker}=d]}|_{n+1}e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} - T_{dc}^{(l,uv)}|_n \quad \text{//Equation (19)} \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1}(W_{cts}^{(l)[\text{ker}=d]}|_n - \eta\cdot\frac{\partial Loss}{\partial\overline{W}_{cts}^{(l)[\text{ker=d}]}})e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} - T_{dc}^{(l,uv)}|_n \\
&= (\sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W_{cts}^{(l)[\text{ker}=d]}|_n e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} - T_{dc}^{(l,uv)}|_n) - \eta\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\frac{\partial Loss}{\partial\overline{W}_{cts}^{(l)[\text{ker=d}]}}e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\
&= -\eta\sum_{t=0}^{K-1}\sum_{s=0}^{K-1}\frac{\partial Loss}{\partial\overline{W}_{cts}^{(l)[\text{ker=d}]}}e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \quad \text{//Equation (19)} \\
&= -\eta MN\frac{\partial Loss}{\partial\overline{T}_{dc}^{(l,uv)}} \quad \text{//Equation (20)}
\end{aligned}$$

Therefore, we prove that any step on $W_{cts}^{(l)[ker=d]}$ equals to MN step on $T_{dc}^{(uv)}$. In this way, pull Equation (25) in the change of $T^{(l,uv)}$ can be computed as follows.

$$\left(\Delta T^{(l,uv)}\right)^\top = -\eta MN\sum_{u'=0}^{M-1}\sum_{v'=0}^{N-1}\chi_{u'v'uv}\left(\overline{\mathbb{T}}^{(u'v')(l-1:1)}\overline{\mathbf{g}}^{(u'v')} + \delta_{u'v'}\overline{\boldsymbol{\beta}}'\right)\frac{\partial Loss}{\partial(\overline{\mathbf{h}}^{(u'v')})^\top}\overline{\mathbb{T}}^{(u'v')(L:l+1)} \tag{26}$$

□

A.4. Proofs of Assumption 4.1 and Theorem 4.2

We prove Assumption 4.1 in the main paper, as follows.

Proof. Given an initialized, cascaded, convolutional decoder network with L convolutional layers, let us focus on the behavior of the decoder network in early epochs of training. We notice that each element in the matrix $T^{(l,uv)}$ is exclusively

determined by the c-th channel of the d-th kenel $W^{(l)[ker=d]}_{c,1:K,1:K} \in \mathbb{R}^{K\times K}$ according to Theorem 3.2. Because parameters in $W^{(l)}$ in the decoder network are set to random noises, we can consider that all elements in $T^{(l,uv)}$ irrelevant to each other, *i.e.*, $\forall d \neq d', c \neq c', T^{(l,uv)}_{dc}$ is irrelevant to $T^{(l,uv)}_{d'c'}$. Similarly, since different layers' parameters $W^{(l)}$ are irrelevant to each other in the initialized decoder network, we can consider that elements in different layers' $T^{(l,uv)}$ irrelevant to each other, *i.e.*, $\forall l \neq l'$, elements in $T^{(l,uv)}$ and elements in $T^{(l',uv)}$ are irrelevant to each other. Moreover, since the early training of a DNN mainly modifies a few parameters according to the lottery ticket hypothesis (Frankle & Carbin, 2018), we can still assume such irrelevant relationships in early epochs, as follows. □

Then, we prove Theorem 4.2, as follows.

Proof. **We first prove that $T^{(l,uv)}_{dc}$ follows a Gaussian distribution of complex numbers.**

According to Assumption 4.1, each convolutional weight follows a Gaussian distribution, *i.e.*, $W^{\text{ker}=d}_{cts} \sim \mathcal{N}(\mu_l, \sigma^2_l)$. For the convenience of proving, let us extend $W^{\text{ker}=d}_{cts}$ into an complex number. In this way, $W^{\text{ker}=d}_{cts}$ follows a Gaussian distribution of complex numbers, *i.e.*, $W^{ker=d}_{cts} \sim Complex\mathcal{N}(\mu_l, \sigma^2_l, 0)$.

Previous studies (Tse & Viswanath, 2005) proved that given N complex numbers, if each complex number follows a Gaussian distribution, then the linear summation of these N complex numbers also follows a Gaussian distribution of complex numbers. Since $T^{(l,uv)}_{dc}$ is a linear combination of $\forall t, s, W^{(l)[\text{ker}=d]}_{cts}$, $T^{(l,uv)}_{dc}$ also follows a Gaussian distribution of complex numbers as follows.

$$\forall d, c \quad T^{(l,uv)}_{dc} \sim \textit{Complex}\mathcal{N}(\hat{\mu}, \hat{\sigma}^2, r)$$

where

$$\begin{aligned}
\mu &= \mathbb{E}[T^{(l,uv)}_{dc}] \quad //\text{By definetion of } \mu \\
&= \mathbb{E}[\sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W^{(l)[\text{ker}=d]}_{cts} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi}] \quad //\text{Equation (19)} \\
&//\forall \mathrm{t} \neq \mathrm{t}' \text{ or } \mathrm{s} \neq \mathrm{s}' : \mathbb{E}[\mathrm{W}^{(\mathrm{l})[\text{ker}=\mathrm{d}]}_{\mathrm{cts}} \mathrm{W}^{(\mathrm{l})[\text{ker}=\mathrm{d}]}_{\mathrm{ct's'}}] = \mathbb{E}[\mathrm{W}^{(\mathrm{l})[\text{ker}=\mathrm{d}]}_{\mathrm{cts}}]\mathbb{E}[\mathrm{W}^{(\mathrm{l})[\text{ker}=\mathrm{d}]}_{\mathrm{ct's'}}] \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} \mathbb{E}[W^{(l)[\text{ker}=d]}_{cts}] e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\
&= \mu_l \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \quad //\mathbb{E}[\mathrm{W}^{(\mathrm{l})[\text{ker}=\mathrm{d}]}_{\mathrm{cts}}] = \mu_{\mathrm{l}} \\
&\quad //Let\ R_{uv} = \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\
&= \mu_l R_{uv}
\end{aligned}$$

$$
\begin{aligned}
\sigma^2 &= \mathbb{E}[(T_{dc}^{(l,uv)} - \mathbb{E}[T_{dc}^{(l,uv)}])\overline{(T_{dc}^{(l,uv)} - \mathbb{E}[T_{dc}^{(l,uv)}])}] \quad \text{//By definetion of } \sigma^2 \\
&= Var[T_{dc}^{(l,uv)}] \\
&= Var[\sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W_{cts}^{(l)[\text{ker}=d]} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi}] \quad \text{//Equation (19)} \\
&\text{//}\forall \text{t} \neq \text{t}' \text{ or s} \neq \text{s}' : \mathbb{E}[\text{W}_{\text{cts}}^{(\text{l})[\text{ker}=\text{d}]}\text{W}_{\text{ct}'\text{s}'}^{(\text{l})[\text{ker}=\text{d}]}] = \mathbb{E}[\text{W}_{\text{cts}}^{(\text{l})[\text{ker}=\text{d}]}]\mathbb{E}[\text{W}_{\text{ct}'\text{s}'}^{(\text{l})[\text{ker}=\text{d}]}] \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} Var[W_{cts}^{(l)[\text{ker}=d]} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi}] \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} Var[W_{cts}^{(l)[\text{ker}=d]}] \quad \text{//} Var[aX] = |a|^2 Var[X] \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} \sigma_l^2 \quad \text{//} Var[W_{cts}^{(l)[\text{ker}=d]}] = \sigma_l^2 \\
&= K^2\sigma_l^2
\end{aligned}
$$

$$
\begin{aligned}
r &= \mathbb{E}[(T_{dc}^{(l,uv)} - \mathbb{E}[T_{dc}^{(l,uv)}])(T_{dc}^{(l,uv)} - \mathbb{E}[T_{dc}^{(l,uv)}])] \quad \text{//By definetion of r} \\
&= C[T_{dc}^{(l,uv)}] \quad \text{//Define C}[\mathbf{X}] = \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])] \\
&= C[\sum_{t=0}^{K-1}\sum_{s=0}^{K-1} W_{cts}^{(l)[\text{ker}=d]} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi}] \quad \text{//Equation (19)} \\
&\text{//}\forall \text{t} \neq \text{t}' \text{ or s} \neq \text{s}' : \mathbb{E}[\text{W}_{\text{cts}}^{(\text{l})[\text{ker}=\text{d}]}\text{W}_{\text{ct}'\text{s}'}^{(\text{l})[\text{ker}=\text{d}]}] = \mathbb{E}[\text{W}_{\text{cts}}^{(\text{l})[\text{ker}=\text{d}]}]\mathbb{E}[\text{W}_{\text{ct}'\text{s}'}^{(\text{l})[\text{ker}=\text{d}]}] \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} C[W_{cts}^{(l)[\text{ker}=d]} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi}] \\
&= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} C[W_{cts}^{(l)[\text{ker}=d]}] e^{i(\frac{2ut}{M}+\frac{2vs}{N})2\pi} \quad \text{//} C[aX] = a^2 C[X] \\
&= \sigma_l^2 \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} e^{i(\frac{2ut}{M}+\frac{2vs}{N})2\pi} \quad \text{//} Var[W_{cts}^{(l)[\text{ker}=d]}] = \sigma_l^2 \\
&= \sigma_l^2 R_{2u,2v} \quad \text{// } R_{uv} = \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi}
\end{aligned}
$$

Finally, let us consider the value of R_{uv}.

$$
\begin{aligned}
R_{uv} &= \sum_{t=0}^{K-1}\sum_{s=0}^{K-1} e^{i(\frac{ut}{M}+\frac{vs}{N})2\pi} \\
&= \sum_{t=0}^{K-1} e^{i(\frac{2u\pi}{M})t} \sum_{s=0}^{K-1} e^{i(\frac{2v\pi}{N})s} \\
&= \frac{\sin(\frac{Ku}{M}\pi)}{\sin(\frac{u}{M}\pi)} \cdot \frac{\sin(\frac{Kv}{N}\pi)}{\sin(\frac{v}{N}\pi)} \cdot e^{i(\frac{(K-1)u}{M}+\frac{(K-1)v}{N})\pi} \quad \text{//According to Equation (13)}
\end{aligned}
$$

Therefore, we prove that $T_{dc}^{(l,uv)}$ **follows a Gaussian distribution of complex numbers.**

$$
\begin{aligned}
&\forall d, c \quad T_{dc}^{(l,uv)} \sim \mathit{Complex}\mathcal{N}(\hat{\mu} = \mu_l R_{uv}, \hat{\sigma}^2 = K^2\sigma_l^2, r = \sigma_l^2 R_{2u,2v}) \\
&\text{s.t. } R_{uv} = \frac{\sin(uK\pi/M)}{\sin(u\pi/M)} \frac{\sin(vK\pi/N)}{\sin(v\pi/N)} e^{i(\frac{(K-1)u}{M}+\frac{(K-1)v}{N})\pi}
\end{aligned} \tag{27}
$$

Then, we prove Theorem 4.2 as follows.

According to Equation (27), $\forall d, c, l : \mathbb{E}[T_{dc}^{(l,uv)}] = \mu_l R_{uv}, Var[T_{dc}^{(l,uv)}] = K^2\sigma_l^2$.

$$\begin{aligned} SOM(T_{dc}^{(l,uv)}) &= \mathbb{E}[|T_{dc}^{(l,uv)}|^2] \\ &= |\mathbb{E}[T_{dc}^{(l,uv)}]|^2 + Var[T_{dc}^{(l,uv)}] \\ &= |\mu_l R_{uv}|^2 + K^2\sigma_l^2 \end{aligned} \tag{28}$$

Then, we have

$$\begin{aligned} \log(SOM(\mathbb{T}^{(uv)(L:1)})) &= \log(\mathbb{E}[|\mathbb{T}^{(uv)(L:1)}|^2]) \\ &= \log(\mathbb{E}[|T^{(L,uv)}\mathbb{T}^{(uv)(L-1:1)}|^2]) \\ &//\text{According to Assumption 4.1, and } \mathrm{C_l} = 1 \\ &= \log(\mathbb{E}[|T^{(L,uv)}|^2]\mathbb{E}[|\mathbb{T}^{(uv)(L-1:1)}|^2]) \\ &= \log((|\mu_L R_{uv}|^2 + K^2\sigma_L^2)SOM(\mathbb{T}^{(uv)(L-1:1)})) \quad //\text{Equation (28)} \\ &= \log(\prod_{l=1}^{L} |\mu_l R_{uv}|^2 + K^2\sigma_l^2) \\ &= \sum_{l=1}^{L} \log(|\mu_l R_{uv}|^2 + K^2\sigma_l^2) \end{aligned}$$

□

For the more general case that each convolutional kernel contains more than one channel, *i.e.*, $\forall l, C_l > 1$, the $SOM(\mathbb{T}^{(uv)(L:1)})$ also approximately exponentially increases along with the depth of the network with a quite complicated analytic solution, as proved below. Note that the following proof is based Assumption 4.1. Besides, we further assume that all elements in $\mathbb{T}^{(uv)(l:1)}$ are independent with each other. *I.e.*, $\forall d \neq d'; c \neq c', \mathbb{E}[\mathbb{T}_{dc}^{(uv)(l:1)}\mathbb{T}_{d'c'}^{(uv)(l:1)}] = \mathbb{E}[\mathbb{T}_{dc}^{(uv)(l:1)}]\mathbb{E}[\mathbb{T}_{d'c'}^{(uv)(l:1)}]$.

Proof. According to Equation (27), all elements in $T^{(l,uv)}$ follow the same Gaussian distribution. Therefore, we have

$$\begin{aligned} \mathbb{E}[T^{(l,uv)}] &= \mathbb{E}[T_{dc}^{(l,uv)}]\mathbf{1}_{(C_l \times C_{l-1})} \\ &= \mu_l R_{uv}\mathbf{1}_{(C_l \times C_{l-1})} \end{aligned} \tag{29}$$

and we have

$$\begin{aligned} SOM(T^{(l,uv)}) &= SOM(T_{dc}^{(l,uv)})\mathbf{1}_{(C_l \times C_{l-1})} \\ &= (|\mu_l R_{uv}|^2 + K^2\sigma_l^2)\mathbf{1}_{(C_l \times C_{l-1})} \end{aligned} \tag{30}$$

Let us first consider the expectation of $\mathbb{T}^{(uv)(L:1)}$ as follows.

$$\begin{aligned} \mathbb{E}[\mathbb{T}^{(uv)(L:1)}] &= \mathbb{E}[T^{(L,uv)}\mathbb{T}^{(uv)(L-1:1)}] \\ &= (C_{L-1}\mathbb{E}[T_{dc}^{(L,uv)}]\mathbb{E}[\mathbb{T}_{dc}^{(uv)(L-1:1)}])\mathbf{1}_{(C_L \times C_0)} \quad //\text{Assumption 4.1, Equation (29)} \\ &= (C_{L-1}\mu_l R_{uv}\mathbb{E}[\mathbb{T}_{dc}^{(uv)(L-1:1)}])\mathbf{1}_{(C_L \times C_0)} \quad //\text{Equation (27)} \\ &= \left(\frac{1}{C_L}\prod_{l=1}^{L} C_l\mu_l R_{uv}\right)\mathbf{1}_{(C_L \times C_0)} \quad //\text{Assumption 4.1} \end{aligned} \tag{31}$$

Then, we have

$$
\begin{aligned}
&SOM(\mathbb{T}^{(uv)(L:1)}) \\
&= \mathbb{E}[|\mathbb{T}^{(uv)(L:1)}|^2] \\
&= \mathbb{E}[|T^{(L,uv)}\mathbb{T}^{(uv)(L-1:1)}|^2] \\
&= (C_{L-1}SOM(T_{dc}^{(L,uv)})SOM(\mathbb{T}_{dc}^{(uv)(L-1:1)}) + C_{L-1}(C_{L-1}-1)|\mathbb{E}[T_{dc}^{(L,uv)}]\mathbb{E}[\mathbb{T}_{dc}^{(uv)(L-1:1)}]|^2)\mathbf{1}_{(C_L\times C_0)} \\
&//\text{According to Assumption 4.1 and Equation (30),} \\
&//\text{we further Assume } \forall \mathrm{d}\neq \mathrm{d}';\mathrm{c}\neq \mathrm{c}', \mathbb{E}[\mathbb{T}_{\mathrm{dc}}^{(\mathrm{uv})(\mathrm{l}:1)}\mathbb{T}_{\mathrm{d'c'}}^{(\mathrm{uv})(\mathrm{l}:1)}] = \mathbb{E}[\mathbb{T}_{\mathrm{dc}}^{(\mathrm{uv})(\mathrm{l}:1)}]\mathbb{E}[\mathbb{T}_{\mathrm{d'c'}}^{(\mathrm{uv})(\mathrm{l}:1)}] \\
&= (C_{L-1}(|\mu_L R_{uv}|^2 + K^2\sigma_L^2)SOM(\mathbb{T}_{dc}^{(uv)(L-1:1)}) + \frac{C_{L-1}-1}{C_{L-1}}|\mathbb{E}[\mathbb{T}_{dc}^{(uv)(L:1)}]|^2)\mathbf{1}_{(C_L\times C_0)} \\
&//\text{According to Equation (28), Equation (31)} \\
&= \left(\frac{1}{C_L}\prod_{l=1}^{L} C_l(|\mu_l R_{u,v}|^2 + (K\sigma_l)^2) + \sum_{l=2}^{L}\frac{C_{l-1}-1}{C_{l-1}}|\frac{1}{C_l}\prod_{k=1}^{l} C_k\mu_k R_{u,v}|^2 \prod_{j=l+1}^{L} C_{j-1}\Big(|\mu_j R_{u,v}|^2 + (K\sigma_j)^2\Big)\right)\mathbf{1}_{C_L\times C_0}
\end{aligned}
\tag{32}
$$

Therefore, we prove that for the more general case that $\forall l, C_l > 1$, the second-order moment $SOM(\mathbb{T}^{(uv)(L:1)})$ also approximately exponentially increases along with the depth of the network.

□

A.5. Proof of Theorem 4.3

In this section, we prove Theorem 4.3 in the main paper, as follows.

Proof.

$$
\begin{aligned}
\mathbb{E}_{F^{(c)}}[G_{uv}^{(c)}] &= \mathbb{E}[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F_{mn}^{(c)} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi}] \quad //\text{Equation (19)} \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\mathbb{E}[F_{mn}^{(c)}]e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi} \\
&= a\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi} \quad //\mathrm{F}_{\mathrm{mn}}^{(\mathrm{c})} \sim \mathcal{N}(\mathrm{a},\sigma^2) \\
&= aMN\delta_{uv}; 0\le u < M, 0\le v < N \quad //\text{Equation (15)}
\end{aligned}
\tag{33}
$$

$$
\begin{aligned}
&\mathbb{E}_{F^{(c)}}[H_{uv}^{(c)}] \\
&= \mathbb{E}_{F^{(c)}}[\sum_{m=0}^{M'-1}\sum_{n=0}^{N'-1} \tilde{F}_{mn}^{(c)} e^{-i(\frac{um}{M'}+\frac{vn}{N'})2\pi}] \quad //\text{Equation (19)} \\
&= \mathbb{E}_{F^{(c)}}[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F_{mn}^{(c)} e^{-i(\frac{um}{M'}+\frac{vn}{N'})2\pi}] \\
&= a\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{-i(\frac{um}{M'}+\frac{vn}{N'})2\pi} \quad //\mathrm{F}_{\mathrm{mn}}^{(\mathrm{c})} \sim \mathcal{N}(\mathrm{a},\sigma^2) \\
&= a\frac{\sin(\frac{Mu}{M'}\pi)}{\sin(\frac{u}{M'}\pi)}\frac{\sin(\frac{Nv}{N'}\pi)}{\sin(\frac{v}{N'}\pi)} e^{-i(\frac{(M-1)u}{M'}+\frac{(N-1)v}{N'})\pi}; 0\le u < M', 0\le v < N' \quad //\text{Equation (13)}
\end{aligned}
\tag{34}
$$

$$
\begin{aligned}
\mathit{Var}_{F^{(c)}}[G_{uv}^{(c)}] &= \mathit{Var}[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F_{mn}^{(c)} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi}] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \mathit{Var}[F_{mn}^{(c)} e^{-i(\frac{um}{M}+\frac{vn}{N})2\pi}] \quad //\forall m,n; F_{mn}^{(c)} \text{ is i.i.d} \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \mathit{Var}[F_{mn}^{(c)}] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \sigma^2 \quad //F_{mn}^{(c)} \sim \mathcal{N}(a,\sigma^2) \\
&= MN\sigma^2
\end{aligned} \tag{35}
$$

$$
\begin{aligned}
\mathit{Var}_{F^{(c)}}[H_{uv}^{(c)}] &= \mathit{Var}[\sum_{m=0}^{M'-1}\sum_{n=0}^{N'-1} \tilde{F}_{mn}^{(c)} e^{-i(\frac{um}{M'}+\frac{vn}{N'})2\pi}] \\
&= \mathit{Var}[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F_{mn}^{(c)} e^{-i(\frac{um}{M'}+\frac{vn}{N'})2\pi}] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \mathit{Var}[F_{mn}^{(c)} e^{-i(\frac{um}{M'}+\frac{vn}{N'})2\pi}] \quad //\forall m,n; F_{mn}^{(c)} \text{ is i.i.d} \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \mathit{Var}[F_{mn}^{(c)}] \\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \sigma^2 \quad //F_{mn}^{(c)} \sim \mathcal{N}(a,\sigma^2) \\
&= MN\sigma^2; 0 \le u < M', 0 \le v < N'
\end{aligned} \tag{36}
$$

When $0 \le u < M, 0 \le v < N$:

$$
\begin{aligned}
\mathit{SOM}(H_{uv}^{(c)}) - \mathit{SOM}(G_{uv}^{(c)}) &= |\mathbb{E}[H_{uv}^{(c)}]|^2 + \mathit{Var}(H_{uv}^{(c)}) - (|\mathbb{E}[G_{uv}^{(c)}]|^2 + \mathit{Var}(G_{uv}^{(c)})) \\
&= (a\tau_{uv})^2 + MN\sigma^2 - (aMN)^2\delta_{uv} - MN\sigma^2 \\
&\quad //\text{According to Equation (33),Equation (34),Equation (35) and Equation (36)} \\
&= (a\tau_{uv})^2 - (aMN)^2\delta_{uv}
\end{aligned} \tag{37}
$$

Therefore, We prove that $\forall 0 \le u < M, 0 \le v < N, u+v \neq 0$

$$
\mathit{SOM}(H_{uv}^{(c)}) - \mathit{SOM}(G_{uv}^{(c)}) = (a\tau_{uv})^2 \tag{38}
$$

When $u = v = 0$:

$$
\mathit{SOM}(H_{uv}^{(c)}) = \mathit{SOM}(G_{uv}^{(c)}) \tag{39}
$$

□

A.6. Proof of Theorem 4.4

In this section, we prove Theorem 4.4 in the main paper, as follows.

Proof.

$$
G_{uv}^{(c)} = \sum_{m=0}^{M_0-1}\sum_{n=0}^{N_0-1} F_{mn}^{(c)} e^{-i(\frac{um}{M_0}+\frac{vn}{N_0})2\pi} \quad //\text{Equation (19)} \tag{40}
$$

$$
\begin{aligned}
H^{(c)}_{u+(s-1)M_0,v+(t-1)N_0} &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \tilde{F}^{(c)}_{mn} e^{-i(\frac{(u+(s-1)M_0)m}{M}+\frac{(v+(t-1)N_0)n}{N})2\pi} \quad //\text{Equation (19)} \\
&= \sum_{m=0}^{M_0-1}\sum_{n=0}^{N_0-1} F^{(c)}_{mn} e^{-i(\frac{(u+(s-1)M_0)(m\cdot ratio)}{M}+\frac{(v+(t-1)N_0)(n\cdot ratio)}{N})2\pi} \\
&= \sum_{m=0}^{M_0-1}\sum_{n=0}^{N_0-1} F^{(c)}_{mn} e^{-i(\frac{(u+(s-1)M_0)m}{M/ratio}+\frac{(v+(t-1)N_0)n}{N/ratio})2\pi} \\
&//M = M_0 \cdot ratio; N = N_0 \cdot ratio \\
&= \sum_{m=0}^{M_0-1}\sum_{n=0}^{N_0-1} F^{(c)}_{mn} e^{-i(\frac{(u+(s-1)M_0)m}{M_0}+\frac{(v+(t-1)N_0)n}{N_0})2\pi} \\
&= \sum_{m=0}^{M_0-1}\sum_{n=0}^{N_0-1} F^{(c)}_{mn} e^{-i(\frac{um}{M_0}+\frac{vn}{N_0})2\pi} \cdot e^{-i((s-1)m+(t-1)n)2\pi} \\
&= \sum_{m=0}^{M_0-1}\sum_{n=0}^{N_0-1} F^{(c)}_{mn} e^{-i(\frac{um}{M_0}+\frac{vn}{N_0})2\pi} \quad //\mathrm{s},\mathrm{t} \in \mathcal{Z} \\
&= G^{(c)}_{uv} \quad //\text{Equation (40)}
\end{aligned}
\tag{41}
$$

Therefore we prove that:

$$
\forall c,u,v, \quad H^{(c)}_{u+(s-1)M_0,v+(t-1)N_0} = G^{(c)}_{uv} \qquad \text{s.t. } s=1,\dots,M/M_0;\ t=1,\dots,N/N_0 \tag{42}
$$

□

A.7. Proof of Theorem 4.5

In this section, we prove Theorem 4.5 in the main paper. Recall that we consider the input $x \in \mathbb{R}^{M\times N}$ with a single channel to simplify the proof. Then, $G \in \mathbb{C}^{M\times N}$, $H \in \mathbb{C}^{M\times N}$, and $H^* \in \mathbb{C}^{M\times N}$ denote spectrums of the input, the output, and the target image to fit, respectively. Specifically, $H^*_{u_1v_1} = (1-A)G^{(u_1v_1)}$, and $H^*_{u_2v_2} = (1+\overline{A})G^{(u_2v_2)}$, where $A = \alpha e^{i\phi}$ and $\overline{A}$ denotes the conjugate of A; $\alpha > 0$; $\phi < \pi/2$. Theorem 4.5 proves a case that weights $\mathbf{W}$ is optimized to satisfy $H_{u_1v_1} = H^*_{u_1v_1}$ and $H_{u_2v_2} = H^*_{u_2v_2}$ simultaneously.

Proof. Let us provide specific constrains as follows, so as to make the auto-encoder learnable (i.e., ensuring $\Delta\mathbf{W}$ is real-valued) and make the objective function can reach zero by a single step of gradient descent.

$$
\lambda_1 = \frac{1}{|G_{u_1v_1}|^2}; \qquad \lambda_2 = \frac{1}{|G_{u_2v_2}|^2} \tag{43}
$$

$$
A = \alpha e^{i\phi}, \quad \textit{s.t. } \phi = (\frac{(K-1)(u_2-u_1)}{M} + \frac{(K-1)(v_2-v_1)}{N})\frac{\pi}{2} \tag{44}
$$

$$
\mathbb{T}^{(u_2v_2)(L:1)} = \mathbb{T}^{(u_1v_1)(L:1)} = 1 \tag{45}
$$

$$
\sum_{l=1}^{L} \left\| \mathbb{T}^{(u_1v_1)(L:l+1)} \right\|^2 \left\| \mathbb{T}^{(u_1v_1)(l-1:1)} \right\|^2 = \sum_{l=1}^{L} \left\| \mathbb{T}^{(u_2v_2)(L:l+1)} \right\|^2 \left\| \mathbb{T}^{(u_2v_2)(l-1:1)} \right\|^2 \tag{46}
$$

If the objective function can reach zero (*i.e.*, $H_{u_1v_1} = H^*_{u_1v_1}$ and $H_{u_2v_2} = H^*_{u_2v_2}$) by a single step of gradient descent, the change of weights $\Delta\mathbf{W}$ can be rewritten as $\eta\frac{\partial Loss}{\partial \mathbf{W}}$, where η denotes the learning rate, $\frac{\partial Loss}{\partial \mathbf{W}}$ denotes the gradient on weights. Then, we will prove the formulations of η and $\frac{\partial Loss}{\partial \mathbf{W}}$.

According to Corollary 3.3, the network output H_{uv} can be computed as follows.

$$
H_{uv} = \mathbb{T}^{(uv)(L:1)} G_{uv}, \tag{47}
$$

where $\mathbb{T}^{(uv)(L:1)} = T^{(L,uv)}T^{(L-1,uv)}\cdots T^{(1,uv)} \in \mathbb{C}^{1\times 1}$.

According to Corollary 3.4, after a single step of gradient descent, the change of $\mathbb{T}^{(uv)(L:1)}$ can be computed as follows.

$$
\begin{aligned}
&\Delta\mathbb{T}^{(uv)(L:1)}\\
&= \Delta(T^{(L,uv)}T^{(L-1,uv)}\cdots T^{(1,uv)})\\
&\approx \sum_{l=1}^{L}\mathbb{T}^{(L:l+1)(uv)}\Delta T^{(l,uv)}\mathbb{T}^{(l-1:1)(uv)} \quad //\text{First order approximation}\\
&= -\eta MN\sum_{l=1}^{L}\mathbb{T}^{(L:l+1)(uv)}(\sum_{u'v'}\chi_{u'v'uv}\overline{\mathbb{T}}^{(l-1:1)(uv)}\overline{G}_{uv}\frac{\partial Loss}{\partial\overline{H}_{uv}^{T}}\mathbb{T}^{(L:l-1)(uv)})^{T}\mathbb{T}^{(l-1:1)(uv)}\\
&= -2\eta MN(\lambda_1\chi_{u_1v_1uv}\overline{G}_{u_1v_1}(H_{u_1v_1}-H^*_{u_1v_1})+\lambda_2\chi_{u_2v_2uv}\overline{G}_{u_2v_2}(H_{u_2v_2}-H^*_{u_2v_2}))\sum_{l=1}^{L}\left\|\mathbb{T}^{(L:l+1)(uv)}\right\|^2\left\|\mathbb{T}^{(l-1:1)(uv)}\right\|^2\\
&//\text{According to Equation (43)}\\
&= -2\eta MN(\chi_{u_1v_1uv}(\mathbb{T}^{(L:1)(u_1v_1)}-\frac{H^*_{u_1v_1}}{G_{u_1v_1}})+\chi_{u_2v_2uv}(\mathbb{T}^{(L:1)(u_2v_2)}-\frac{H^*_{u_2v_2}}{G_{u_2v_2}}))\sum_{l=1}^{L}\left\|\mathbb{T}^{(L:l+1)(uv)}\right\|^2\left\|\mathbb{T}^{(l-1:1)(uv)}\right\|^2\\
&= -2\eta MN(A\chi_{u_1v_1uv}-\overline{A}\chi_{u_2v_2uv})\sum_{l=1}^{L}\left\|\mathbb{T}^{(L:l+1)(uv)}\right\|^2\left\|\mathbb{T}^{(l-1:1)(uv)}\right\|^2 //\text{Equation (45)}
\end{aligned}
\tag{48}
$$

For frequencies $[u_1, v_1]$ and $[u_2, v_2]$, we have:

$$
\Delta\mathbb{T}^{(L:1)(u_1v_1)} = -2\eta MN(A\chi_{u_1v_1u_1v_1}-\overline{A}\chi_{u_2v_2u_1v_1})\sum_{l=1}^{L}\left\|\mathbb{T}^{(L:l+1)(u_1v_1)}\right\|^2\left\|\mathbb{T}^{(l-1:1)(u_1v_1)}\right\|^2 \tag{49}
$$

$$
\Delta\mathbb{T}^{(L:1)(u_2v_2)} = -2\eta MN(A\chi_{u_1v_1u_2v_2}-\overline{A}\chi_{u_2v_2u_2v_2})\sum_{l=1}^{L}\left\|\mathbb{T}^{(L:l+1)(u_2v_2)}\right\|^2\left\|\mathbb{T}^{(l-1:1)(u_2v_2)}\right\|^2 \tag{50}
$$

For any other frequency component $[u, v]$, where $u \neq u_1, u \neq u_2, v \neq v_1, v \neq v_2$, the change $\Delta\mathbb{T}^{(uv)(L:1)}$ can be cpmputed as the linear combination of the $\Delta\mathbb{T}^{(L:1)(u_1v_1)}$ and $\Delta\mathbb{T}^{(L:1)(u_2v_2)}$ as follows.

$$
\Delta\mathbb{T}^{(uv)(L:1)} = a_1\Delta\mathbb{T}^{(L:1)(u_1v_1)} + a_2\Delta\mathbb{T}^{(L:1)(u_2v_2)} \tag{51}
$$

where $a_1 \in \mathbb{C}$ and $a_2 \in \mathbb{C}$ are two complex coefficients, which keep unchanged during the learning process.

On the other hand, the exact change of $\mathbb{T}^{(uv)(L:1)}$ can be directly computed given the objective function. For frequencies $[u_1, v_1]$ and $[u_2, v_2]$, we have:

$$
\Delta\mathbb{T}^{(L:1)(u_1v_1)} = \frac{H^*_{u_1v_1}}{G_{u_1v_1}} - \mathbb{T}^{(L:1)(u_1v_1)} = -A \tag{52}
$$

$$
\Delta\mathbb{T}^{(L:1)(u_2v_2)} = \frac{H^*_{u_2v_2}}{G_{u_2v_2}} - \mathbb{T}^{(L:1)(u_2v_2)} = \overline{A} \tag{53}
$$

For any other frequency component $[u, v]$, the change $\Delta\mathbb{T}^{(uv)(L:1)}$ can be computed as follows:

$$
\Delta\mathbb{T}^{(uv)(L:1)} = -a_1A + a_2\overline{A} \tag{54}
$$

Then, combining Equation (51) and Equation (54), we can obtain the value of η.

$$\begin{aligned}\eta &\propto \frac{-a_1A+a_2\overline{A}}{-a_1(A\chi_{u_1v_1u_1v_1}-\overline{A}\chi_{u_2v_2u_1v_1})-a_2(A\chi_{u_1v_1u_2v_2}-\overline{A}\chi_{u_2v_2u_2v_2})} \quad \text{//Equation (46)}\\ &= \frac{-a_1A+a_2\overline{A}}{-a_1A(\chi_{u_1v_1u_1v_1}-e^{-i2\phi}\chi_{u_2v_2u_1v_1})+a_2\overline{A}(\chi_{u_2v_2u_2v_2}-e^{i2\phi}\chi_{u_1v_1u_2v_2})}\\ &= \frac{-a_1A+a_2\overline{A}}{-a_1A+a_2\overline{A}}\cdot\frac{MN}{K^2-\frac{\sin(K(u_2-u_1)\pi/M)\sin(K(v_2-v_1)\pi/N)}{\sin((u_2-u_1)\pi/M)\sin((v_2-v_1)\pi/N)}}\\ &= \frac{MN}{K^2-\frac{\sin(K(u_2-u_1)\pi/M)\sin(K(v_2-v_1)\pi/N)}{\sin((u_2-u_1)\pi/M)\sin((v_2-v_1)\pi/N)}}\end{aligned} \tag{55}$$

And $\left\|\frac{Loss}{\partial\mathbf{W}}\right\|$ can be computed as follows.

$$\begin{aligned}\left\|\frac{Loss}{\partial\mathbf{W}}\right\|^2 &= \sum_{l,d,c}\sum_{t,s}(\frac{\partial Loss}{\partial\overline{\mathbf{W}}^{(l)[\text{ker=d}]}_{cts}})^2\\ &= \sum_{l,d,c}\sum_{t,s}\frac{\partial Loss}{\partial\overline{\mathbf{W}}^{(l)[\text{ker=d}]}_{cts}}\cdot\frac{\partial Loss}{\partial\overline{\mathbf{W}}^{(l)[\text{ker=d}]}_{cts}}\\ &= \sum_{l,d,c}\sum_{t,s}\frac{\partial Loss}{\partial\overline{\mathbf{W}}^{(l)[\text{ker=d}]}_{cts}}\sum_{u,v}\frac{\partial Loss}{\partial\overline{T}^{(l,uv)}_{dc}}e^{-i(\frac{ut}{M}+\frac{vs}{N})2\pi}\\ &= \sum_{l,d,c}\sum_{u,v}\frac{\partial Loss}{\partial\overline{T}^{(l,uv)}_{dc}}\sum_{t,s}\frac{\partial Loss}{\partial\overline{\mathbf{W}}^{(l)[\text{ker=d}]}_{cts}}e^{-i(\frac{ut}{M}+\frac{vs}{N})2\pi}\\ &= \sum_{l,d,c}\sum_{u,v}\frac{\partial Loss}{\partial\overline{T}^{(l,uv)}_{dc}}\frac{\partial Loss}{\partial T^{(l,uv)}_{dc}}\\ &= \sum_{l,d,c}\sum_{u,v}\frac{\partial Loss}{\partial\overline{T}^{(l,uv)}_{dc}}\frac{\partial Loss}{\partial T^{(l,uv)}_{dc}}\\ &= \sum_{l,d,c}\sum_{u,v}|\frac{\partial Loss}{\partial\overline{T}^{(l,uv)}_{dc}}|^2\\ &\propto \sum_{l,d,c}\sum_{u,v}\alpha^2\\ &\propto \alpha^2\end{aligned} \tag{56}$$

Therefore, the weight change $\|\Delta\mathbf{W}\|$ can be computed as follows:

$$\begin{aligned}\|\Delta W\| &= \eta\cdot\left\|\frac{Loss}{\partial W}\right\|\\ &\propto \eta\cdot\sqrt{\|\Delta W\|^2}\\ &\propto \frac{MN\alpha}{K^2-\frac{\sin(K(u_2-u_1)\pi/M)\sin(K(v_2-v_1)\pi/N)}{\sin((u_2-u_1)\pi/M)\sin((v_2-v_1)\pi/N)}} \quad \text{//Equation (55) and Equation (56)}\end{aligned} \tag{57}$$

□

B. Discussions about different factors that weakening high-frequency components

B.1. Effects of the network depth

If the decoder network is deep, then the decoder network is less likely to learn high-frequency components. It is because $|R_{uv}|$ is relatively large for low-frequency components. In this way, the large effect of a single layer's $T^{(l,uv)}$ of low-frequency components on $\log SOM(\mathbb{T}^{(uv)(L:1)})$, *i.e.*, $\log(|\mu_l R_{uv}|^2+K^2\sigma_l^2)$, can be accumulated through different layers according to

(1) the Law of Large Numbers, and (2) the independent effects $\log(|\mu_l R_{uv}|^2 + K^2\sigma_l^2)$ between different layers' $T^{(l,uv)}$ on $\log \textit{SOM}(\mathbb{T}^{(uv)(L:1)}) = \sum_{l=1}^{L} \log(|\mu_l R_{uv}|^2 + K^2\sigma_l^2)$.

Therefore, the large $|R_{u^{\text{low}}v^{\text{low}}}|$ value for a low-frequency component $[u^{\text{low}}, v^{\text{low}}]$ makes $\mathbb{T}^{(u^{\text{low}}v^{\text{low}})(L:1)}$ more likely to have a large norm, whereas the small $|R_{u^{\text{high}}v^{\text{high}}}|$ value for a high-frequency component $[u^{\text{high}}, v^{\text{high}}]$ makes $\mathbb{T}^{(u^{\text{high}}v^{\text{high}})(L:1)}$ less likely to have a large norm. This indicates that **a deep decoder network will almost certainly strengthen the encoding of low-frequency components of the input sample, while weaken the encoding of high-frequency components.**

B.2. Effects of the initialization of network parameters

If the expectation μ_l of convolutional weights in each l-th layer has a large absolute value $|\mu_l|$, then the decoder network is less likely to learn high-frequency components. It is because according to Theorem 4.2, a large absolute value $|\mu_l|$ boosts the imbalance effects $|\mu_l R_{uv}|^2$ among different frequency components, thereby strengthening the trend of encoding low-frequency components of the input sample.

B.3. Effects of the convolutional kernel size

If the convolutional kernel size K is small, then the decoder network is less likely to learn high-frequency components. It is because according to Theorem 4.2, a large K value alleviates imbalance of the second-order moment $\textit{SOM}(\mathbb{T}^{(uv)(L:1)})$ between low frequencies and high frequencies caused by the imbalance of $|R_{uv}|$. Thus, a small K value strengthens the trend of encoding low-frequency components of the input sample.

B.4. Effects of the distribution of the training data

If the cascaded convolutional decoder network is trained on natural images, then the decoder network is less likely to learn high-frequency components. Previous studies (Ruderman, 1994) have empirically found that natural images were dominated by low-frequency components. Specifically, frequency spectrums of natural images follow a Power-law distribution. *I.e.*, low-frequency components (*e.g.*, the frequency component $[u, v]$ closed to $[0, 0]$, $[0, N-1]$, $[M-1, 0]$, and $[M-1, N-1]$) have much larger length $\|\mathbf{g}^{(uv)}\|_2 = \sqrt{\sum_c |G_{uv}^{(c)}|^2}$ than other frequency components. Besides, according to rules of the forward propagation in Equation (4) and the change of $T^{(l,uv)}$ in Equation (5), if the frequency component $\mathbf{g}^{(uv)}$ of the input image has a large magnitude, then $\mathbf{h}^{(uv)}$ of the output image also has a large magnitude. This means that using natural images as the input strengthens the trend of encoding low-frequency components.

C. More experimental results

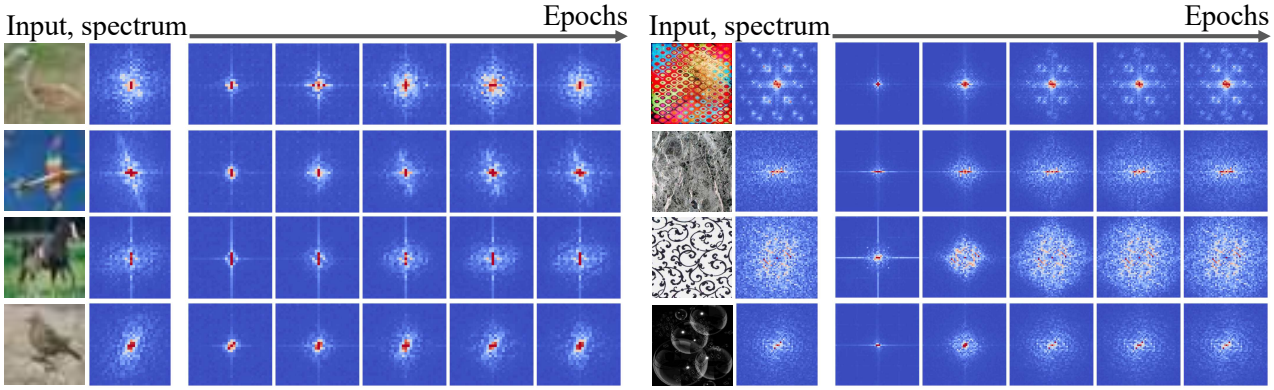
C.1. Verifying that a neural network usually learned low-frequent components first.

In section, we provide more experimental results to verify that a neural network usually learned low-frequent components first, which had already been shown in Figure 1(a) in the main paper. Here, we also constructed a cascaded convolutional auto-encoder by using the VGG-16 as the encoder network. The decoder network contained three upconvolutional layers for the CIFAR-10 dataset, and contained three upconvolutional layers for the Broden dataset. Each convolutional/upconvolutional layer in the auto-encoder applied zero-paddings and was followed by a batch normalization layer and an ReLU layer. The auto-encoder was trained using the mean squared error (MSE) loss for image reconstruction. Results in Figure 6 verified that the auto-encoder usually learned low-frequent components first and gradually learned higher frequecies. We also attached the generated image below its spectrum map in Figure 7, in order to help people understand the learning process of the auto-encoder.

C.2. Verifying that the upsampling operation made a decoder network repeat strong signals at certain frequencies of the generated image.

In section, we provide more experimental results to verify that the upsampling operation in the decoder repeats strong frequency components of the input to generate spectrums of upper layers.

First, we conducted experiments to verify Theorem 4.4 in the main paper, which claims that the upsampling operation repeats the strong magnitude of the fundamental frequency $G_{00}^{(c)}$ of the lower layer to different frequency components $\forall c, H_{u^*v^*}^{(c)}$ of the higher layer, where $u^* = 0, M_0, 2M_0, 3M_0, \ldots; v^* = 0, N_0, 2N_0, 3N_0, \ldots$. To verify this, given an image, let



(a) CIFAR-10 dataset (b) Broden dataset

Figure 6. Magnitude maps of feature spectrums of different epochs' network output. Each magnitude map was averaged over all channels. For clarity, we moved low frequencies to the center of the spectrum map, and moved high frequencies to corners of the spectrum map. Note that we set the magnitude of the fundamental frequency to be the same with the magnitude of the second significant frequency. For resutls in (b), we only visualized components in the center of the spectrum map with the range of relatively low frequencies $u \in \{u|0 \leq u < M/8\} \cup \{u|7M/8 \leq u < M\}; v \in \{v|0 \leq v < N/8\} \cup \{v|7N/8 \leq v < N\}$ for clarity.

the image pass through four cascaded upsampling layers. We visualized the feature spectrum generated by each upsampling layer, in order to verify whether the upsampling operation repeated the strong magnitude of the fundamental frequency of the input image to different frequency components of the feature spectrum generated by upsampling layers. Results on the CIFAR-10 dataset and the Tiny-ImageNet dataset in Figure 8 verified Theorem 4.4.

Second, we provide more results on real neural networks, which have already been shown in Figure 1(b) in the main paper. We also constructed a cascaded convolutional auto-encoder by using the VGG-16 as the encoder network. The decoder network contained four upconvolutional layers. Each convolutional/upconvolutional layer in the auto-encoder applied zero-paddings and was followed by a batch normalization layer and an ReLU layer. The auto-encoder was trained on the Broden dataset using the mean squared error (MSE) loss for image reconstruction. Results in Figure 9 verified Theorem 4.4.

C.3. Verifying that the zero-padding operation strengthened the encoding of low-frequency components.

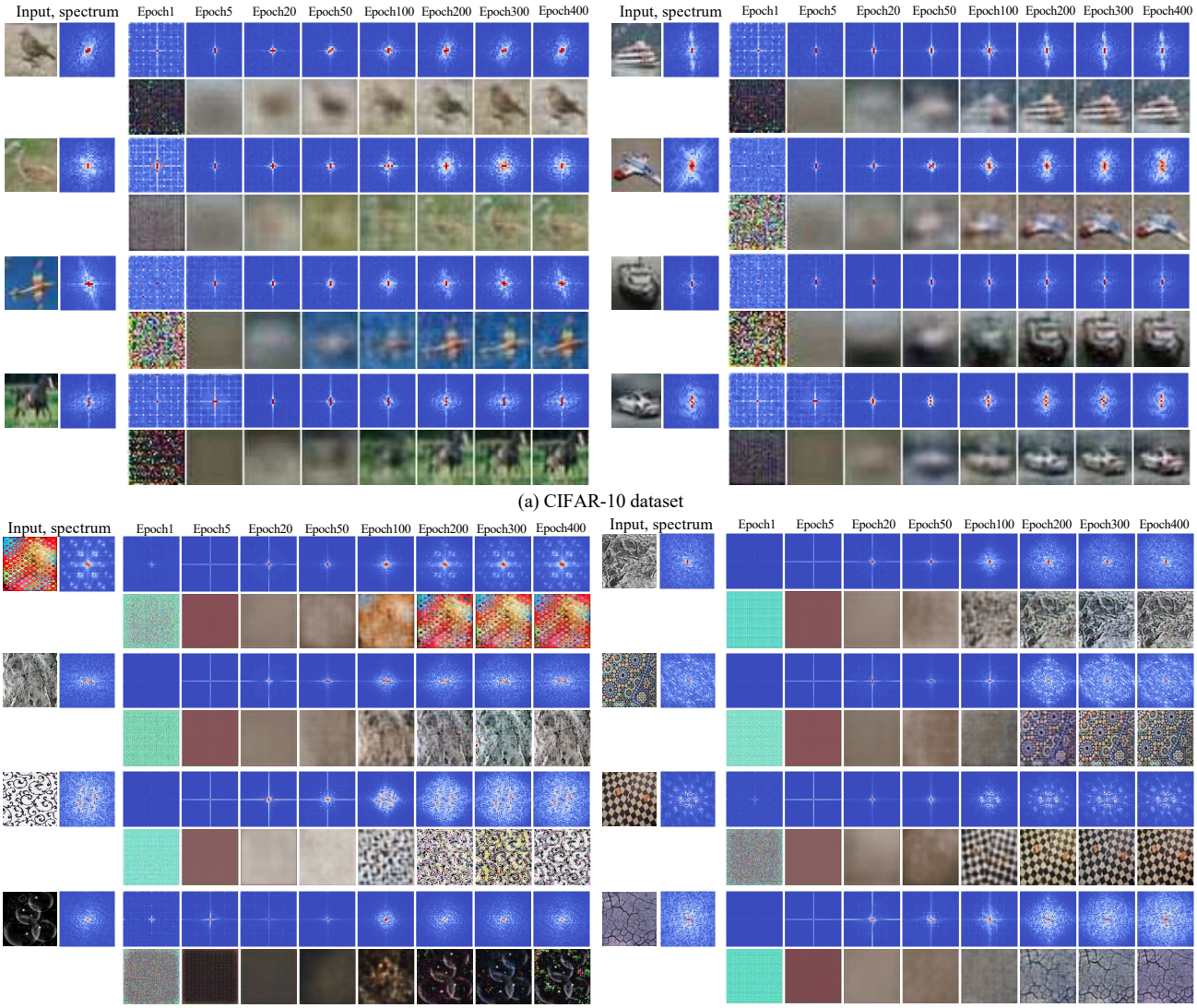
In section, we provide more experimental results to verify that the zero-padding operation strengthened the encoding of low-frequency components, which had already been shown in Figure 4(c) in the main paper. Here, we also constructed the following three baseline networks. The first baseline network contained 5 convolutional layers, and each layer applied zero-paddings. Each convolutional layer contained 16 convolutional kernels (kernel size was 7×7), except for the last layer containing 3 convolutional kernels. The second baseline network and the third baseline network were constructed by replacing all zero-padding operations with circular padding operations and replacing all zero-padding operations with mirror padding operations, respectively. Results in Figure 10 verified that the zero-padding operation strengthened the encoding of low-frequency components.

C.4. Verifying that a deep network strengthened low-frequency components.

In section, we provide more experimental results to verify that a deep network strengthened low-frequency components, which had already been shown in Figure 4(a) in the main paper. Here, we also constructed a network with 50 convolutional layers. Each convolutional layer applied zero-paddings to avoid changing the size of feature maps, and was followed by an ReLU layer. We visualized feature spectrums of different convolutional layers. Results on the CIFAR-10 dataset and the Tiny-ImageNet dataset in Figure 11 show that magnitudes of low-frequency components increased along with the network layer number.

C.5. Verifying that a larger absolute mean value μ_l of each l-th layer's parameters strengthened low-frequency components.

In section, we provide more experimental results to verify that a larger absolute mean value μ_l of each l-th layer's parameters strengthened low-frequency components, which had already been shown in Figure 4(b) in the main paper. Here, we also



(a) CIFAR-10 dataset

(b) Broden dataset

Figure 7. Magnitude maps of feature spectrums and the corresponding generated images of different epochs. Results show that in the very few epochs of the training, the network removed noisy signal caused by the upsampling, to some extent, which were in the grid pattern in the spectrum. After that, the network learned low-frequency components first, and then gradually learned higher frequencies. Each magnitude map in this figure was averaged over all channels. For clarity, we moved low frequencies to the center of the spectrum map, and moved high frequencies to corners of the spectrum map. Note that we set the magnitude of the fundamental frequency to be the same with the frequency that had the second large magnitude.

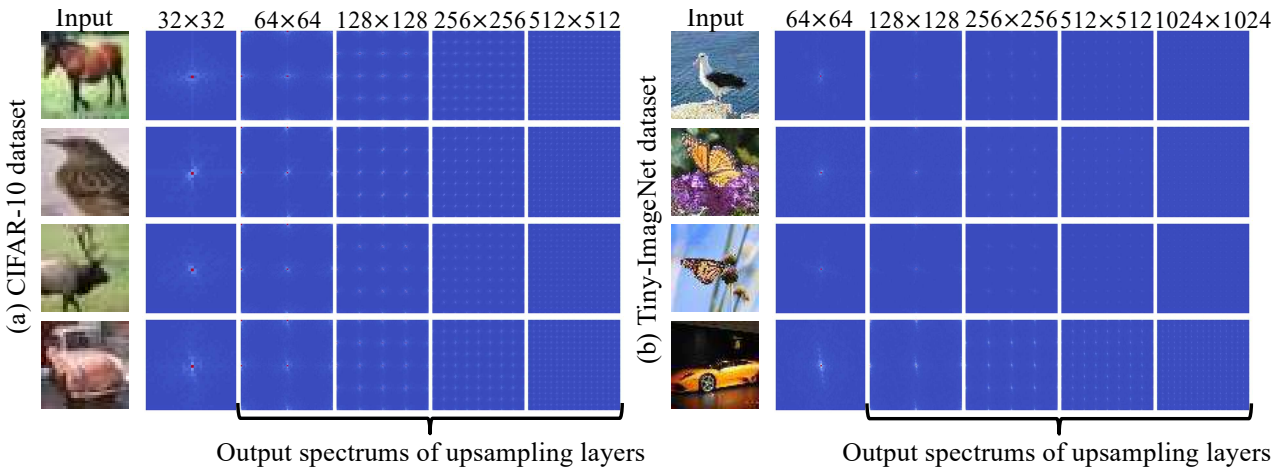


Figure 8. Magnitude maps of feature spectrums after one/two/there/four upsampling layers. Each magnitude map was averaged over all channels. For clarity, we moved low frequencies to the center of the spectrum map, and moved high frequencies to corners of the spectrum map.

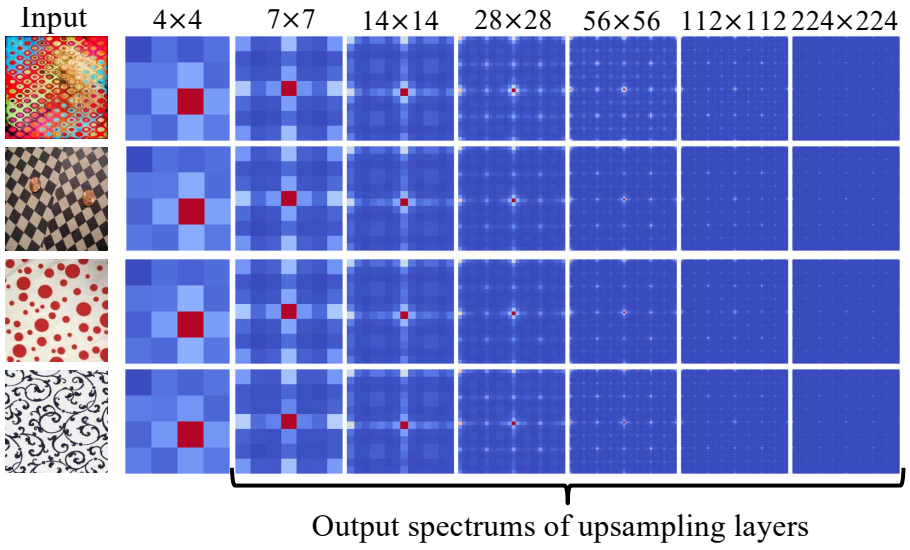


Figure 9. Magnitude maps of feature spectrums after one/two/there/four/five/six upsampling layers. Each magnitude map was averaged over all channels. For clarity, we moved low frequencies to the center of the spectrum map, and moved high frequencies to corners of the spectrum map.

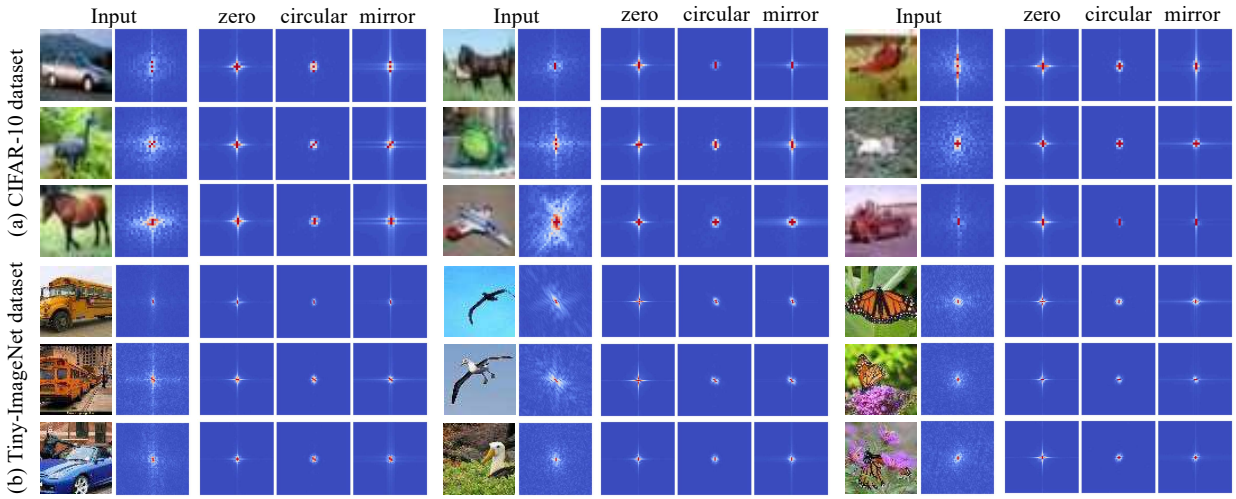


Figure 10. A network with zero-padding operations usually strengthened more low-frequency components than a network with circular padding operations and a network with mirror padding operations. Here, each magnitude map of the feature spectrum was averaged over all channels. For clarity, we move low frequencies to the center of the spectrum map, move high frequencies to corners of the spectrum map, and set the magnitude of the fundamental frequency to be the same with the frequency that has the second large magnitude.

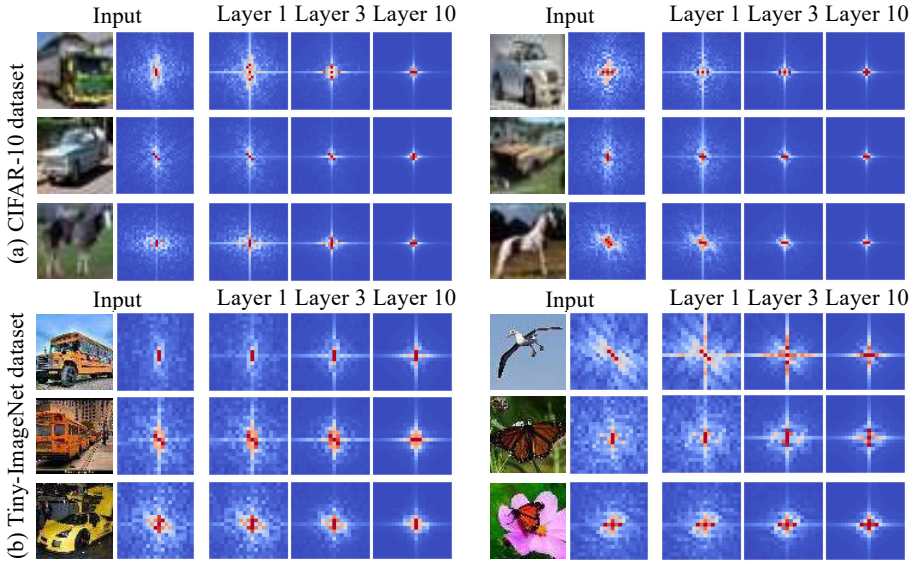


Figure 11. Comparing feature spectrums of different layers. Results show that higher layers of a network usually generated features with more low-frequency components. For clarity, we move low frequencies to the center of the spectrum map, move high frequencies to corners of the spectrum map, and set the magnitude of the fundamental frequency to be the same with the frequency that has the second large magnitude. For resutls in (b), we only visualized components in the center of the spectrum map with the range of relatively low frequencies $u \in \{u|0 \leq u < M/6\} \cup \{u|5M/6 \leq u < M\}; v \in \{v|0 \leq v < N/6\} \cup \{v|5N/6 \leq v < N\}$ for clarity.

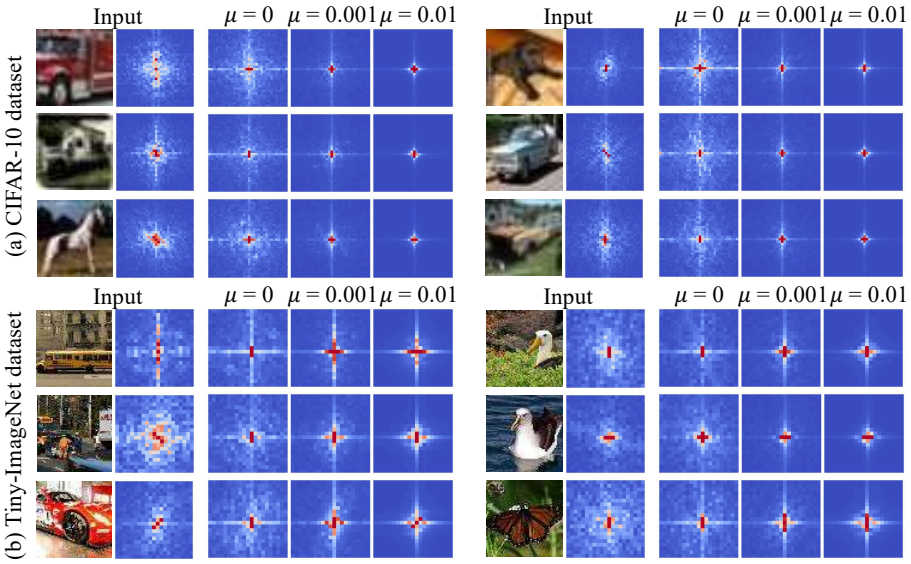


Figure 12. A network whose convolutional weights had a mean value significantly biased from 0 usually strengthened low-frequency components, but weakened high-frequency components. Here, each magnitude map of the feature spectrum was averaged over all channels. For clarity, we moved low frequencies to the center of the spectrum map, moved high frequencies to corners of the spectrum map, and set the magnitude of the fundamental frequency to be the same with the frequency that has the second large magnitude. For resutls in (b), we only visualized components in the center of the spectrum map with the range of relatively low frequencies $u \in \{u|0 \leq u < M/6\} \cup \{u|5M/6 \leq u < M\}; v \in \{v|0 \leq v < N/6\} \cup \{v|5N/6 \leq v < N\}$ for clarity.

applied a network architecture with 5 convolutional layers. Each layer contained 16 convolutional kernels (kernel size was 9×9), except for the last layer containing 3 convolutional kernels. Based on this architecture, we constructed three networks, whose parameters were sampled from Gaussian distributions $\mathcal{N}(\mu = 0, \sigma^2 = 0.01^2)$, $\mathcal{N}(\mu = 0.001, \sigma^2 = 0.01^2)$, and $\mathcal{N}(\mu = 0.01, \sigma^2 = 0.01^2)$, respectively. Results on the CIFAR-10 dataset and the Tiny-ImageNet dataset in Figure 12 show that magnitudes of low-frequency components increased along with the absolute mean value of parameters.

C.6. Discussions on the curse of dimension

In this section, we discuss the problem of the curse of dimension, when we compute the cosine similarity of two high-dimensional vectors with as many as 32^2, 64^2 or 224^2 dimensions. In general, for each pair of extremely high-dimensional vectors, it's hard for these vectors to have very high cosine similarity. It is because even if the noisy differences between many pairs of dimensions can be ignored, in the process of calculating the sum of squares, these noisy differences will accumulate. Therefore, the cosine similarity of two extremely high-dimensional vectors will not be particularly large.

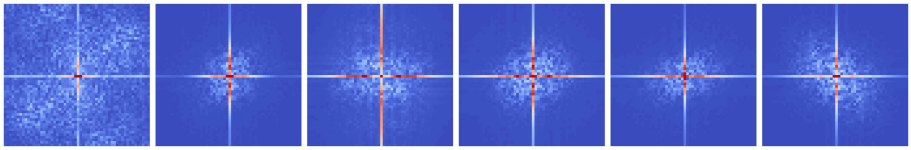
C.7. Verifying that Assumption 3.1 can be applied to fully trained DNNs

Assumption 4.1 shows that in early training of a DNN, all elements in $T^{(l,uv)}$ are irrelevant to each other, and $\forall l \neq l'$, elements in $T^{(l,uv)}$ and $T^{(l',uv)}$ are irrelevant to each other. In this section, we further conducted experiments to verify that such irrelevant relationships also existed in a fully trained DNN. To this end, we constructed a cascaded convolutional auto-encoder by using the VGG-16 as the encoder network. The decoder network contained ten convolutional layers, where the 1st, 3rd, 5th, 7th, and 9th layers were traditional convolutional layers, and the 2nd, 4th, 6th, 8th, and 10th layers were upconvolutional layers. Each convolutional/upconvolutional layer applied zero-paddings and was followed by a batch normalization layer and an ReLU layer. The network was trained on the Tiny-ImageNet dataset using the mean squared error (MSE) loss for image reconstruction.

Let the above auto-encoder be trained to convergence. Then, we computed the Pearson's correlation coefficient between each random pair of variables $|\Delta T^{(l,uv)}_{d_1c_1}|$ and $|\Delta T^{(l,uv)}_{d_2c_2}|$ through different images, denoted by $r(|\Delta T^{(l,uv)}_{d_1c_1}|, |\Delta T^{(l,uv)}_{d_2c_2}|)$, to measure the relevance between two elements $T^{(l,uv)}_{d_1c_1}$ and $T^{(l,uv)}_{d_2c_2}$ in $T^{(l,uv)}$. Here, $\Delta T^{(l,uv)}$ denoted the change of $T^{(l,uv)}$ when we updated parameters $\mathbf{W}$ for a single gradient-descent step on a single input sample. Results in Table 1 show that even when the network was fully trained, different elements in $T^{(l,uv)}$ had low Pearson's correlation coefficient r. This proved

Table 1. Pearson's correlation coefficient between each random pair of variables $|\Delta T^{(l,uv)}_{d_1c_1}|$ and $|\Delta T^{(l,uv)}_{d_2c_2}|$ through different images.

Depth of the decoder network	$\mathbb{E}_{u,v,d_1,c_1,d_2,c_2}[r(\lvert\Delta T^{(l,uv)}\_d_1c_1\rvert, \lvert\Delta T^{(l,uv)}\_d_2c_2\rvert)]$
1	0.011
2	-0.002
3	0.018
4	-0.001
5	-0.013
6	0.026
7	0.018



(a) (b1) (b2) (b3) (b4) (b5)

Figure 13. (a) The spectrum generated by the sixth decoder, whose parameters in all layers had zero mean. (b1-b5) Spectrums generated by the first five decoders, whose parameters in a certain layer (the 1st, 2nd, 3rd, 4th, and 5th, respectively) had a large absolute mean value. Results show that no matter which layer of the network had a larger absolute mean value of parameters, there was no significant difference in weakening the encoding of high-frequency components.

that our assumption that all elements in $T^{(l,uv)}$ were irrelevant to each other was reasonable. The last three convolutional layers in the decoder showed a larger Pearson's correlation coefficient, because network parameters close to the output layer had been converged to the principle feature direction of each category. Nevertheless, our experiments showed that for most layers, we could keep Assumption 4.1, which enabled to us to prove that convolution operations in these layers weakened high-frequency components.

C.8. Effects of large absolute mean values on layers with different depth

We conducted experiments to measure the effects of large absolute mean values on layers with different depth. To this end, we compared spectrums of output features, when we set large absolute mean values for parameters in different convolutional layers. Therefore, we constructed five convolutional networks with the same architecture for comparison. Each convolutional network had five convolutional layers. To construct the l-th network for comparison, we sampled parameters of the l-th convolutional layer from the Gaussian distribution $\mathcal{N}(\mu = 0.1, \sigma^2 = 0.1^2)$, and sampled parameters of the remaining four convolutional layers from the Gaussian distribution $\mathcal{N}(\mu = 0, \sigma^2 = 0.1^2)$. Besides, based on this architecture, we also constructed the sixth network by let parameters of all layers be sampled from the Gaussian distribution $\mathcal{N}(\mu = 0, \sigma^2 = 0.1^2)$.

Figure 13 shows results on the Broden dataset. Compared with the sixth network that all parameters had zero mean (see Figure 13(a)), all other networks (see Figure 13(b1-b5)), whose parameters in a certain layer had a large absolute mean value, weakened high-frequency components. Besides, no matter which layer had parameters of a large absolute mean value, there was no significant difference between the five networks in weakening the encoding of high-frequency components.

C.9. Details about the frequency shift

In this section, we provide examples to introduce details about how to shift each salient frequency component $[u, v]$ in the input x to $[u + \Delta u, v]$ or $[u - \Delta u, v]$ towards higher frequencies, as Figure 14 shows. Note that we move low frequencies to the center of the spectrum map, and move high frequencies to the corners of the spectrum map for clarity.

C.10. More experimental details

Table 2 reports the number of epochs for the training of each model and its fitting error $\mathbb{E}_x[\frac{\|x-\hat{x}\|_2^2}{N}]$, where N denoted the number of pixels in the image.

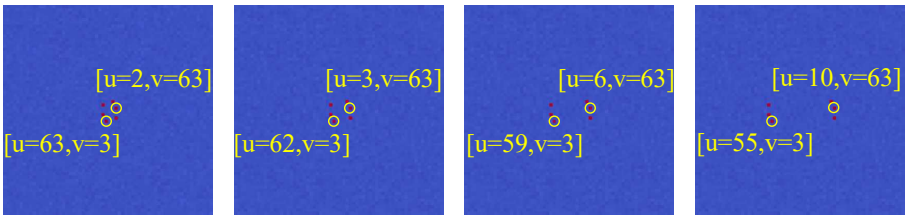


Figure 14. Examples of shifting each salient frequency component $[u, v]$ in the input to $[u + \Delta u, v]$ or $[u - \Delta u, v]$ towards higher frequencies, where $\Delta u = 1, 4, 8$. For clarity, we move low frequencies to the center of the spectrum map, and move high frequencies to the corners of the spectrum map.

Table 2. Number of epochs for the training of each model and its fitting error.

	# training epoch	fitting error
The model used in verifying that a neural network usually learned low frequent components first	400	9.35e-3
The model used in verifying the repeat of certain frequencies.	10	6.97e-2