electronics

MDPI

Article

Deep Learning Based on Fourier Convolutional Neural Network Incorporating Random Kernels

Yuna Han and Byung-Woo Hong *

Computer Science Department, Chung-Ang University, Seoul 156-756, Korea; yuna@image.cau.ac.kr
* Correspondence: hong@cau.ac.kr

Abstract: In recent years, convolutional neural networks have been studied in the Fourier domain for a limited environment, where competitive results can be expected for conventional image classification tasks in the spatial domain. We present a novel efficient Fourier convolutional neural network, where a new activation function is used, the additional shift Fourier transformation process is eliminated, and the number of learnable parameters is reduced. First, the Phase Rectified Linear Unit (PhaseReLU) is proposed, which is equivalent to the Rectified Linear Unit (ReLU) in the spatial domain. Second, in the proposed Fourier network, the shift Fourier transform is removed since the process is inessential for training. Lastly, we introduce two ways of reducing the number of weight parameters in the Fourier network. The basic method is to use a three-by-three sized kernel instead of five-by-five in our proposed Fourier convolutional neural network. We use the random kernel in our efficient Fourier convolutional neural network, whose standard deviation of the Gaussian distribution is used as a weight parameter. In other words, since only two scalars for each imaginary and real component per channel are required, a very small number of parameters is applied compressively. Therefore, as a result of experimenting in shallow networks, such as LeNet-3 and LeNet-5, our method achieves competitive accuracy with conventional convolutional neural networks while dramatically reducing the number of parameters. Furthermore, our proposed Fourier network, using a basic three-by-three kernel, mostly performs with higher accuracy than traditional convolutional neural networks in shallow and deep neural networks. Our experiments represent that presented kernel methods have the potential to be applied in all architecture based on convolutional neural networks.

Keywords: random kernel; Fourier convolutional neural network; image classification

Citation: Han, Y.; Hong, B.-W. Deep Learning Based on Fourier Convolutional Neural Network Incorporating Random Kernels. *Electronics* **2021**, *10*, 2004. https://doi.org/10.3390/electronics10162004

Academic Editor: Miad Faezipour

Received: 15 July 2021
Accepted: 15 August 2021
Published: 19 August 2021

1. Introduction

The convolutional neural network (CNN) is the most basic neural network based on solving problems of various machine learning tasks, such as classification [1], segmentation, and denoising in computer vision. One of the problems with CNN training is that the convolution operation of all convolutional layers requires considerable cost. In particular, as the size of the image or kernel increases, the amount of computation inevitably increases, resulting in a latency of learning. One method proposed to solve this problem is to change the domain through Fourier transform, and construct a CNN in the frequency domain because the convolution operation in the spatial domain is the same as the point-wise multiplication in the Fourier domain. In general, point-by-point multiplication is more uncomplicated and computationally cheaper to compute than convolution. Prior approaches have focused on improving computational speed to handle the time cost problem [2–4].

There are two factors in determining computational complexity. One is the time complexity that was implemented in existing studies, and the other is the memory complexity that was studied for model weight reduction in the spatial domain [5–10]. However, previous studies on the Fourier domain were not conducted on a method of reducing the

number of parameters that directly affect the complexity of memory. The efficient use of memory is a critical issue since unlimited resources are not provided in the real world. A recent example is an application used in a mobile device that requires a high speed and a lightweight model. In addition, since the number of GPUs or memory is limited, building a neural network that can learn sufficiently with a small or few GPU is important. In addition, CNNs are used in a variety of engineering areas for practical applications, such as conditions monitoring of marine vehicles [11] and fault diagnosis of the cerebral cortex [12]. Therefore, in order to propose a method to perform deep learning in a limited environment in further work, we propose the method of reducing the number of parameters of the convolutional neural network, which is the base of the neural network. Furthermore, previous studies on CNNs in the Fourier domain have focused on the study of efficient neural networks for high-speed training, and the problem of reducing the number of parameters was actively investigated in the spatial domain rather than the Fourier domain; therefore, the goal is to design an efficient CNN by applying kernel methods with a few parameters.

In previous studies, the implementation and pooling method of the convolutional layer corresponding to the spatial domain was actively investigated in the Fourier domain [13]. While the ReLU-based activation function is known to be effective in the spatial domain, an appropriate activation function in the Fourier domain has not been established. The previous approach mainly relies on the approximation of ReLU. However, it is limited in that it has a huge computational cost, cannot function as a nonlinear operation, and cannot expect a higher, or the same, accuracy as ReLU in the spatial domain [14–16]. On the contrary, we present the activation function used in the Fourier domain, which performs the same operation as ReLU in the spatial domain, and introduces a new Fourier convolutional network that applies a new activation function for the Fourier domain. The novel activation function in the Fourier domain is built upon the characteristics of the Fourier transformed image consisting of phase and magnitude, which will be covered in detail in Section 3.

There are two major studies on weight reduction in the Fourier CNN. The first is to adjust the kernel size in our proposed Fourier CNN, and the second is to learn the standard deviation (std.) for the Gaussian distribution by creating a random kernel based on compression sensing. First, unlike the spatial domain with local information, Fourier transformed images have global information. In the spatial domain, a large-sized kernel can be used by finding the location and information of the pixel locally to extract the characteristics of the image. On the contrary, in the Fourier domain, it is expected that even a small-sized kernel will be able to sufficiently identify the characteristics of an image and perform classification tasks by using global information consisting of low- and high-frequency components. Second, in compression sensing, a random vector is generated by multiplying the sparse signal by a random matrix of Gaussian distribution to compressively restore the original signal. According to this theory, it is assumed that using a random filter can learn scalar values for a random matrix of a fixed Gaussian distribution, and therefore, by learning a standard for a Gaussian distribution, image classification can be performed with a few parameters.

In conclusion, the contributions of the paper are listed as follows: we present a new activation function in the Fourier domain, discard the unnecessary shift in the Fourier transform process, introduce the novel convolutional neural network, using a small-sized kernel in the Fourier domain based on our proposed activation function, and investigate an efficient convolutional neural network based on the random kernel in the Fourier domain to reduce the number of weight parameters.

2. Related Work

Convolutional neural networks (CNNs) have been used to extract and learn image features from the deep neural networks in computer vision such as classification, segmentation, etc. [1]. In addition, deep neural networks, such as AlexNet, VGG, DeseNet and

ResNet [17–20], have been widely developed and have been effectively applied to various tasks using large datasets, such as ImageNet [20].

However, deep neural network models have a problem that the number of learning parameters increases as the layer becomes deeper. Therefore, significant memory costs for learning a model and high computational costs are required.

In particular, it is difficult to apply in a limited environment of mobile devices with limited hardware resources. As one of the methods to solve this problem, model compression [6,8] has been explored. For example, ShuffleNets are designed to optimize for the mobile device environment [7]; MobileNet-based models are described to reduce the number of parameters through depth-wise separable convolution [5,9]; and SqueezeNet shows similar performance to AlexNet with fewer parameters by reducing the number of input channels in the 3×3 filter and replacing the 3×3 filter with the 1×1 filter [10].

Other techniques for model compression include pruning [21,22], distillation [23] and quantization [24]. First, pruning is a method of removing neuron or weights with less important information [21,22]. Second, distillation refers to a mechanism of transferring the knowledge of a more extensive ensembled neural network to a relatively small single neural network in order to solve the inefficient use of memory resources that generally occurs when a model is ensembled [23]. Third, quantization is a technique of minimizing the loss of accuracy versus full precision while using a low bit width [24].

However, these methods have a fundamental limitation in that they cannot solve the time computational problem of convolution of the image and the kernel in the convolutional layer, which is the stage of learning features of the image. In recent years, to solve the time–cost problem in a convolutional layer, a CNN in the Fourier domain through Fourier transform was actively studied, using the theory that convolution in the spatial domain is equivalent to point-wise multiplication in the Fourier domain [14,16,25–27].

The convolutional layer of the Fourier domain for time complexity has been widely explored because point-by-point multiplication in the Fourier domain is much faster than convolution in the spatial domain. Furthermore, one of the leading fast Fourier transform techniques is based on discrete Fourier transform (DFT). In general, the Cooley–Tukey algorithm is used for the fast Fourier algorithm. However, training the convolutional layer in the Fourier domain requires an additional operation—inverse Fourier transform. Therefore, pooling and activation functions in the frequency domain for fully training the CNNs in the frequency domain have been studied for many recent years [2–4].

First, truncating the low-frequency components of the Fourier transformed image into a predetermined size, which are used as a spectral representation and for extracting only important information, has been proposed as a method of implementing spectral pooling. However, because the Fourier transform is performed before spectral pooling and the inverse Fourier transform is used after each pooling, there is an additional computational cost to implement the iteration. Moreover, the proposed method has not considered the process of training the convolutional layer in the Fourier domain and has not examined the problem of computational cost [13]. Another proposed method of pooling is discrete Fourier transform (DFT) based magnitude pooling. The first component of the Fourier transformed image is the DC component. DC stands for direct current in electrical engineering, but it simply refers to the zero frequency or the mean value of the frequencies in the Fourier domain. The whole process of training is implemented by calculating the magnitude of the DFT and reducing the resolution to include the first component from the values obtained. However, the phase information is not considered in the DFT-based pooling method, even though preserving both the phase and magnitude of the Fourier transformed image is vital for reconstructing the image after inverse Fourier transform. In addition, using a number of parameters for creating ensemble networks is difficult to regard as an efficient network in the frequency domain [28].

Second, suitable activation functions in the Fourier domain have been actively investigated. Research on the activation function in the Fourier domain is largely divided into two main directions. One is to roughly estimate the activation function in the Fourier

domain, which has a shape similar to the activation function in the spatial domain [15,16]. The other is to present a new formula for the Fourier-based activation function, taking into account the properties of the frequency components [14]. One of the most popular functions is spectral ReLU (SReLU) [15], which is designed to approximate the conventional ReLU function in terms of a quadratic function. The basic idea of SReLU is to find a quadratic function that is determined to be roughly similar to ReLU in the spatial domain. However, calculating the quadratic function for each activation function has considerable time complexity [29]. Especially when the input size is large or the layer depth is deep, performing the activation function by SReLU causes a computational burden.

Another approach to implementing the approximation function in the Fourier domain is to find a linear function similar to the tanh and sigmoid functions in the spatial domain, using the linearity property of Fourier transform. However, the presented linear functions cannot perform as nonlinear functions, making it tough to train the complex model.

One of the recent approaches of an activation function is using the property of low- and high-frequency components in the Fourier domain. For instance, a second harmonics superposition activation function (2SReLU) has been proposed to overlap the first and second harmonics, including the DC component of the Fourier transformed image. Since the first harmonic of the image contains low frequencies and the second harmonic of the image has some high frequencies, the neural network can be trained at both low and high frequencies. In addition, because the Fourier transformed image is composed of complex numbers, it is expressed as a magnitude value of the real and imaginary parts of the image plus periodic functions, such as cosine and sine functions, respectively. Considering the composition of the Fourier transformed image function, adding several harmonics causes the sin wave function to converge to zero, like the negative part of ReLU [14]. According to Equation (1), F refers to the Fourier transform, each h_i is the ith harmonic or interval, and hyper-parameters alpha and beta are predetermined to 0.7 and 0.3. After multiplying each alpha and beta for the first and second harmonic weight, respectively, two harmonics are added as follows:

$$F(h_1) \leftarrow \alpha F(h_1) + \beta F(h_2) \tag{1}$$

Yet, in terms of measuring the accuracy of the classification task, 2SReLU [14] is poorly fit to Fourier-based CNNs, compared to the previous activation function, SReLU [15]. Therefore, our novel activation function is focused on the activation function that fits the Fourier domain, while considering the characteristics of the frequency domain image.

$$modReLU(z) = ReLU(|z| + b)e^{i\theta} \tag{2}$$

$$= \begin{cases} (|z| + b)\frac{z}{|z|} & if\ |z| + b \geq 0, \\ 0 & otherwise, \end{cases} \tag{3}$$

Recently, several complex value-based activation functions, such as *modReLU* [30], zReLU [31], and complex *ReLU* ($\mathbb{C}ReLU$ [32]) were introduced. First, modReLU is defined as Equation (3), and it refers that the activation is applied when a learnable bias term b is positive, where z is a complex number and the phase of z is denoted as θ_z. The equation is designed to preserve the pre-activated phase information [30].

Second, $zReLU$ is described as Equation (4). The equation refers that $zReLU$ maintains the input number z when the phase exists in the first quadrant; otherwise, it replaces it with 0 [31].

$$zReLU(z) = \begin{cases} z & if\ \theta_z \in [0, \pi/2], \\ 0 & otherwise, \end{cases} \tag{4}$$

Third, $\mathbb{C}ReLU$ is the latest complex number-based activation function that obtains more information than *modReLU* and $zReLU$, which is explained as Equation (5).

$$\mathbb{C}ReLU(z) = ReLU(\Re(z)) + iReLU(\Im(z)). \tag{5}$$

Assuming that the positive and negative values of the complex-valued image are represented in the four quadrants, $\mathbb{C}ReLU$ has the advantage that information can be obtained from the remaining three quadrants, except when both real and imaginary components are less than 0 since $\mathbb{C}ReLU$ applies $ReLU$ to the real and imaginary components, respectively. On the other hand, in the case of *modReLU*, due to the learning bias term b, a circle with a radius of length b is inactive, and the outside of the circle is active. *zReLU* is active only when the input phase is in the first quadrant.

Furthermore, Fourier-based CNNs were researched to complete training entirely in the frequency domain before entering through fully connected layers [25–27] to eliminate additional computations when performing inverse Fourier transform after applying activation [16] or pooling layers [15,33]. However, the previous training method of CNNs within the Fourier domain has several limitations. First, the activation function of the Fourier domain corresponding to ReLU in the conventional CNN in the spatial domain has not yet been explicitly described [25–27,33]. Second, the existing research on reducing memory cost in the Fourier domain was not conducted in the entire Fourier domain. In addition, the previously presented tanh-based activation function in the spectral domain performs on a different principal from ReLU in conventional CNN [33]. Third, earlier studies on memory cost only considered zero sparsity, using model compression. However, applying a kernel with a small number of parameters can be another efficient training process and has the advantage that it can be applied to various CNNs, regardless of the architecture, without being limited to a compressed model. Therefore, we intend to design an efficient CNN in the Fourier domain by reducing the number of parameters, using different kernel methods that directly affect energy reduction in the memory aspect.

3. Method

3.1. Representation of Random Kernel

3.1.1. Compressed Sensing

Compressed sensing was used to reconstruct original signals and images with a small number of samples, and was developed from traditional Shannon–Nyquist sampling to convert an analog signal to a digital signal. The Shannon–Nyquist theorem is defined as the original signal being taken at least twice as much as the highest frequency samples. In other words, reconstruction is available only when sampling is acquired by satisfying the Nyquist sampling rate. Otherwise, it is not easy to restore the original data since aliasing occurs. In general, data obtained from nature are measured with a device, such as an analog-to-digital converter (ADC), to convert the data from nature to digital. Only a few data are extracted, as the data scan speed can be slow, or the machine can be expensive.

On the other hand, samples in limited environments can be reconstructed through compression detection with samples that are less than the Nyquist sampling rate [34]. The most basic purpose of this theory is to compress and sample the model by converting the under-determined system into an over-determined system, using sparsity. In general, the scarcity of data is found by some raw data with a small amount of information, such as sound, video, and image. In addition, sparse signals can be measured when the original data are transferred to a specific domain. According to this principle, when the original signal to be restored is x and the measured signal is Y, the signal processing of compression sensing can be defined as shown in Equation (6) [34].

$$Y = \phi x \tag{6}$$

As shown in Equation (6), x is a signal vector of size $N \times 1$, and Y is a signal vector of size $M \times 1$. According to the theory of compression sensing, the condition of $N \leq M$ must be satisfied. Additionally, M can be defined as a much smaller amount of data than N. The main point of Equation (6) is to find the M-by-N matrix to transform the under-determined system into an over-determined one. The measurement matrix ϕ, such as an ADC device, is generally determined by a random distribution, such as Gaussian [35]. In addition, the original signal x can be expressed as a product of a sparse signal and a

sensing matrix, as shown in Equation (7), where s is a sparse vector of size $N \times 1$, and ψ is a matrix of size $N \times N$.

$$x = \psi s \tag{7}$$

In other words, Equation (7) refers that s is the coefficient on a ψ basis. Y is newly summarized as shown in Equation (10) by Equations (8) and (9). In other words, Y can be expressed as the product of Θ, which is a sensing matrix of size $M \times N$, and a sparse vector s.

$$Y = \phi x \tag{8}$$

$$= \phi \psi s \tag{9}$$

$$= \Theta s \tag{10}$$

The property of Θ, a sensing matrix, which is also called a reconstruction matrix, is determined by the definition of the restricted isometry property (RIP) of Θ, and RIP is expressed by Equation (11).

$$(1 - \delta_I)\|s\|_2^2 \leq \|\Theta s\|_2^2 \leq (1 + \delta_I)\|s\|_2^2 \tag{11}$$

The isometry constant $\delta_I (I > 0)$ in Equation (11) is the smallest constant value, and a suitable sensing matrix can be acquired because sparse signal s is almost preserved when δ_I is near zero, and the left and right terms of the two inequalities approach each other. By finding I that satisfies these conditions, we can obtain a suitable sensing matrix that can be restored. In particular, Θ should be a random sensing matrix for satisfying the RIP property; therefore, it is important to have a randomly distributed matrix for Θ [34]. The reconstruction process of compressed sensing is shown as Figure 1, which is inspired by [34]. According to Figure 1, the measurement vector Y is a vector of size $M \times 1$, which is the result of the product of the $M \times N$ measurement matrix ϕ and $N \times 1$ original signal x, where $M \leq N$. The original signal x can also be represented by the product of the $N \times N$ random representation matrix and $N \times 1$ sparse vector s. According to these two formulas, the Y vector can be expressed as the multiplication of the $M \times N$ random matrix $\Theta (= \phi\psi)$ and the $N \times 1$ sized sparse vector s.

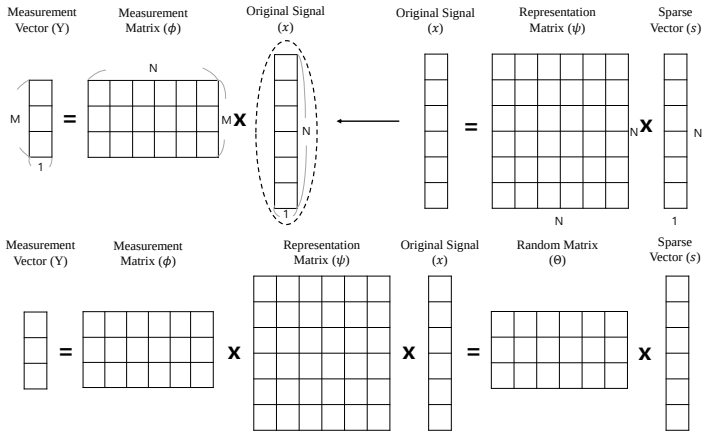


Figure 1. The reconstruction process of the interest signal x in compressed sensing. The purpose of reconstruction is to obtain a measurement vector Y using the measurement matrix and the original signal. Calculating the top of the figure at once can be explained as shown in the figure below.

3.1.2. Random Kernel

The traditional method of learning an image classification task is to train a neural network through convolutional neural networks (CNNs) in the spatial domain. Within a large category, CNNs are divided into convolutional layers for learning the features of an image, and a classifier determines the classes of images called fully connected layers.

The convolutional layer is formed by the convolutional operation of an input image and a kernel composed of learnable parameters. The convolutional layer based on the existing LeNet-5 model [1] proposes a kernel of 5×5 size; for the AlexNet model [20], kernels of various sizes, such as 11×11, 5×5, and 3×3, are presented. In other words, the LeNet-5 model requires 25 trainable parameters per channel, and the AlexNet model needs 121, 25, or 9 parameters, depending on the kernel size. Since the number of learnable parameters increases in proportion to the size of the kernel, and many weight parameters affect the expensive memory cost, various methods of reducing the number of parameters in CNN-based models are actively investigated [5,9,10,21–24].

Many architectures have been proposed to reduce the number of parameters of CNNs in the spatial domain, but there is a limitation, as the number of parameters used in the convolutional layer is different depending on the architecture of each proposed model [5,9,10]. For example, if the k is the size of the kernel, considering that the number of parameters is determined by (batch size) $\times$ (the number of kernel) $\times$ (the number of input channel) $\times$ $(K_h \times K_w)$, the number of weight parameters is primarily affected by the size, and the number of parameters can increase rapidly as the kernel grows. Therefore, we propose a random kernel similar to a random vector that reconstructs the original signal by multiplying a random matrix and a sparse vector in compressed sensing. Our random kernel sets the kernel size, one of the main factors determining the number of parameters, to be the same for all architectures so that the same performance as the conventional CNN kernel can be achieved with fewer parameters. As shown at the top of Figure 2, conventional kernels have (kernel height x kernel width) number of parameters, for example, the sizes of 3×3, 5×5 and 7×7 kernels, in turn, have 9, 25 and 49 learnable parameters, respectively. On the contrary, our proposed random kernel only has one learnable parameter denoted as α, regardless of the size of the kernel; therefore, all 3×3, 5×5 and 7×7 sized kernels have the same number of parameters in a single channel, according to the bottom of Figure 2. In addition, the random kernel consists of the product of a single trainable scalar and a untrainable fixed random matrix initialized with a Gaussian distribution. Therefore, the number of parameters of the random kernel can be expressed as (batch size) $\times$ (the number of kernels) $\times$ (the number of input channel) $\times$ (α). In other words, the learning parameter for each convolutional layer is determined by the number of kernels and the number of input channels. In conclusion, we can expect the advantage of the random kernel to be as follows: first, a random kernel reduces the number of parameters exponentially regardless of the size of the kernel or the depth of the neural network's layers; second, the random kernel can be applied to any types of CNN based models; third, we can potentially expect this to be a method for a light-weight model, performing with cheap memory costs.

3.2. Fourier Convolutional Neural Network

The convolutional layer of convolution-based neural network in the spatial domain is configured to learn the features of an image through a convolution operation between an image and a kernel. However, the convolution operation has a disadvantage in that the calculation is complicated because the input image is multiplied and added by rolling a window as large as the kernel size.

When the image in the spatial domain is transformed into the frequency domain through the Fourier transform, the computational cost is reduced because a simple operator called point-wise multiplication is used instead of the convolution operation. Assuming that functions f and g of x are given in the spatial domain, and Fourier transformed F and G are the functions of X in the frequency domain, Equation (12) is described as follows:

$$f(x) * g(x) \xrightarrow{\mathcal{F}} F(X) \times G(X) \tag{12}$$

For example, given input $x \in \Re^{h \times w}$ and kernel $k \in \Re^{k \times l}$ in the spatial domain, the time complexity is expected as $\mathcal{O}(hwkl)$. On the other hand, $F(x) \in \mathbb{C}^{M \times N}$ with fast Fourier

transform incurs a computation cost of $\mathcal{O}(mnlog(mn))$. Recently, in order to train image classification tasks in the Fourier domain, finding an adequate activation function [14,15,36] and sub-sampling operation [13] in the Fourier domain equivalent to ReLU or max-pooling in the spatial domain was widely explored. However, one of the limitations of the existing studies on Fourier-based activation functions is that the activation functions do not exactly match ReLU.

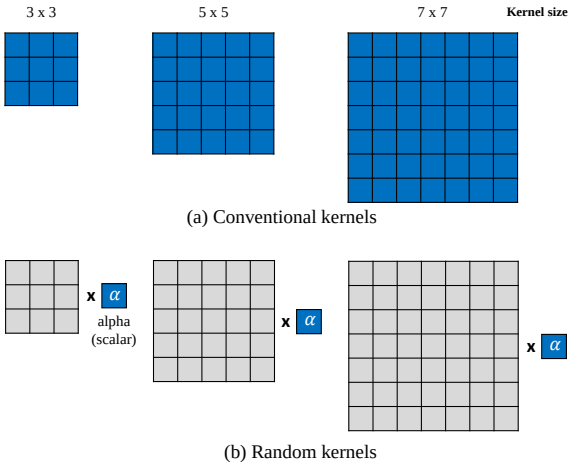


Figure 2. Comparison of the number of parameters between the conventional and our proposed random kernel in the spatial domain. **Blue**: trainable weight parameters. **Gray**: non-trainable weight parameters. (**a**) Conventional kernels. (**b**) Random kernels.

Therefore, this section introduces a Fourier-based activation function that performs the same behavior as ReLU in the spatial domain, and presents a convolutional neural network for fully training in the Fourier domain. Moreover, we remove unnecessary computations during the process of reducing additional computational costs and Fourier transform. First, low- and high-frequency components of all Fourier-transformed images are used without shift Fourier transform operation. Second, the process to Fourier transform the kernel, which is used in the traditional convolutional neural network, is omitted by using the property that the Fourier-transformed Gaussian is also Gaussian. Lastly, by applying the random kernel discussed in Section 3.1.2 to a Fourier-based convolutional neural network, we aim to build an effective convolutional neural network with similar accuracy to conventional spatial convolutional neural networks with few parameters.

3.2.1. Fourier Convolutional Layer

Given any number $x \in \Re$, Fourier transform $f : \Re \xrightarrow{\mathcal{F}} \mathbb{C}$ is defined as in Equation (13).

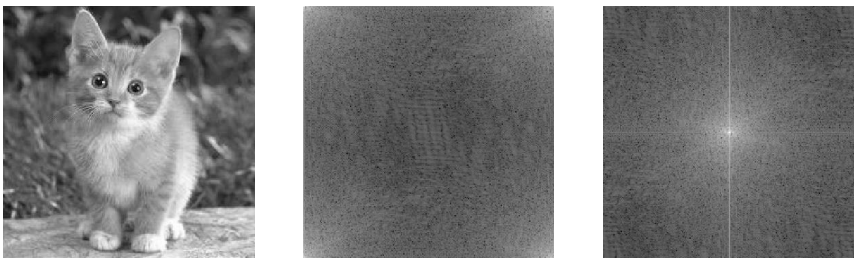
$$\mathcal{F}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx \tag{13}$$

If the input image in a limited range of (channel) $\times$ (height) $\times$ (width) is 2D-Fourier transformed, where $\forall H \in \{0, ..., M-1\}, \forall W \in \{0, ..., N-1\}$, $F(x) \in \mathbb{C}^{M \times N}$ can be expressed as in Equation (14).

$$\mathcal{F}(x)_{H \times W} = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{mn} e^{-2\pi i (\frac{mH}{M} + \frac{nW}{N})} \tag{14}$$

The standard practice has rearranged the frequency components in previous work by shifting the low-frequency portion to the image center as shown on the right side of Figure 3. The first component of the Fourier transformed image represents the zero frequency or the DC component representing the mean in the frequency domain, which is also relocated to the center of the image. Unlike the traditional method, we eliminate the

additional operation by removing the shift operation. Without the Fourier shift operation, the input image is first channel-wise Fourier transformed, and each low-frequency and high-frequency component is at the edge and center of the image, as described in the center image of Figure 3. In addition, the DC component is located at the top left corner.



(a) Image (b) FT non-shift (c) FT shift

Figure 3. (**a**) Original gray-scale image. (**b**) Our proposed Fourier transformed image without shift operation. (**c**) Traditional Fourier transformed image with shift operation.

The converted input $F(x)$ performs an element-wise multiplication for each channel with the kernel $F(k) \in \mathbb{C}^{M \times N}$ of the same size as the input x. Given input image $x \in \mathbb{C}^{height \times width}$ and kernel $k \in \Re^{k \times l}$, the most basic technique of the kernel's Fourier transform is to apply the Fourier transform after zero-padding at the right and bottom of the kernel with size (width—l) and (height—k), respectively.

Our proposed Gaussian random kernel fulfills two purposes. The first is to use a random kernel that learns only one alpha parameter per channel, as in the method introduced in Section 3.1.2. The second is to reduce the computation cost by removing the two steps of zero-padding and Fourier transform, using the property that the Gaussian function is also a Gaussian function even after the Fourier transform. Our proposed random kernel is the same as multiplying the Gaussian distributed random matrix by the alpha representing the distribution scale, so it learns the standard deviation (std.) of the Gaussian distribution in the Fourier domain.

Since the Fourier transformed image $\mathcal{F}(x)$ is complex-valued, the random kernel K should also be initialized to a complex number of the same size. In the case of a convolutional layer based on a random kernel as shown in Figure 4, each real and imaginary component of the random matrix is initialized to the Gaussian distribution, respectively. In practice, however, point-wise multiplication in complex-valued domain is performed as Equation (16). According to the multiplication of the rectangular form of complex number, given $\mathcal{F}(x) = A + Bi$ and $K = C + Di$ ($\mathcal{F}(x) \in \mathbb{C}$,K$\in \mathbb{C}$) is defined as follows:

In addition, standard deviation α is initialized to $\alpha = \alpha_{\Re} + \alpha_{\Im} i$ for complex-valued scalar multiplication for the result of the product of $\mathcal{F}(x)$ and K. As illustrated in Figure 4, our presenting convolutional layer based on the random kernel is expressed as the following Equation (15).

$$(\mathcal{F}(I) \times K) \times \alpha = \alpha_{\Re}(AC - BD) + \alpha_{\Im}(AD + BC)i \tag{15}$$

Practically, our proposed method of implementing the convolutional layer in the Fourier domain is divided in two ways, as shown in Figure 4. One is the random kernel-based convolutional layer, and the other is the convolutional layer with the 3×3 sized kernel regardless of the size of the conventional kernel used for the convolutional layer in the spatial domain.

$$\mathcal{F}(x) \times K = (AC - BD) + (AD + BC)i \tag{16}$$

3.2.2. Fourier Activation Function: PhaseReLU

The standard convolutional neural networks (CNNs) convert the affine function to nonlinear after the activation function, and thus help to construct more complex and various types of models by increasing the learning capacity. The most stable and widely used activation function in the spatial domain is the ReLU function [37]. However, the corre-

sponding activation function in the Fourier domain is not officially established. In particular, as a result of comparing the officially proposed complex-valued activation function, $\mathbb{C}ReLU$, zReLU with ReLU for the baseline and the state-of-the-art Fourier activation function 2SReLU, they all perform poorly. Therefore, we propose a new Fourier-transformed activation function, PhaseReLU—suitable to substitute for ReLU.

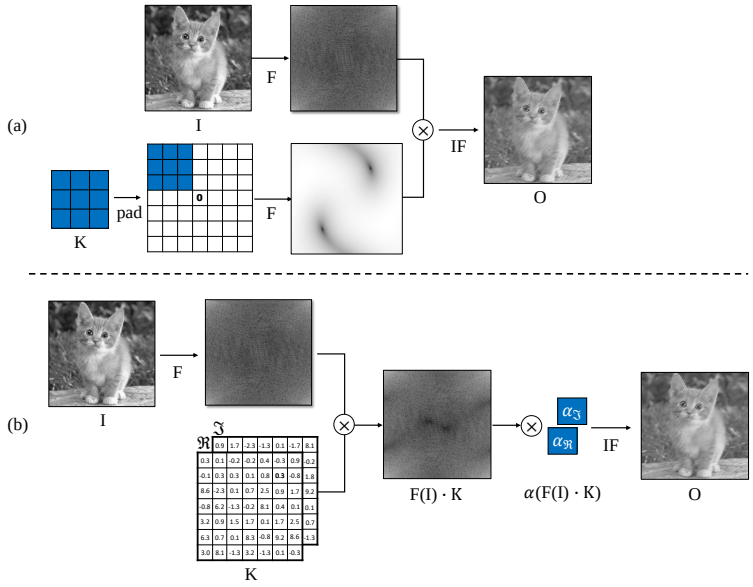


Figure 4. Reconstruction images of our proposed Fourier convolutional layer with a 3×3 sized kernel and efficient Fourier convolutional layer based on random kernel. (**a**) Fourier convolutional layer. (**b**) Efficient Fourier convolutional layer. The input gray-scale image denoted as I and K indicates kernel.

The Fourier transformed image z can be expressed in rectangular form and polar form, described in Equations (17) and (18), respectively.

$$z = a + bi \tag{17}$$

$$z = |z|(\cos\phi + i\sin\phi) = |z|e^{i\phi} \tag{18}$$

In addition, the Fourier transformed image x is composed of phase $\angle x$ or ϕ, and magnitude $|x|$. Let us assume that ϕ is a phase of the image; then, each $a \in \Re$ and $b \in \Im$ is also expressed as $\cos\phi$ and $\sin\phi$, respectively. According to Equations (19) and (20), each component of the magnitude and phase in the polar form is described as follows:

$$|z| = \sqrt{a^2 + b^2} \tag{19}$$

$$\phi = \tan^{-1}\left(\frac{b}{a}\right) \tag{20}$$

Since the magnitude is equal to the square root of multiplying real and imaginary components, it is always positive. In the case of the phase, which is denoted as $\phi_z \in (-\pi, \pi]$, it can be negative or positive, depending on the given complex value. Using these properties, we first take the feature map received from the convolutional layer, and decompose it into the phase and magnitude. We only consider when the phase is $\phi_z \in (-\frac{\pi}{2}, \frac{\pi}{2}]$, so we can expect $z > 0$. Therefore, same as the ReLU, the original values are preserved when the given complex value is positive and the values are replaced with zeros for the negative. In order to pass the activation map to the pooling process, the phase and magnitude should be recomposed after the previous step. Our newly proposed activation function in the frequency domain, PhaseReLU, is proposed in Equation (21). Our implementation of comparing the accuracy for the PhaseReLU to the existing activation functions, such as

CReLU, zReLU, 2SReLU and baseline ReLU with Fashion-MNIST datasets on LeNet-3, is shown in Table 1. As a result, PhaseReLu can replace ReLU, as it can perform classification tasks better than baseline ReLU and other previous activation functions.

$$PhaseReLU(z) = |z|(\cos(ReLU(\phi_z)) + i\sin(ReLU(\phi_z))) \quad (21)$$

Table 1. The accuracy (%) of PhaseReLU and other existing complex-valued activation functions on Fashion-MNIST from LeNet-3.

	ReLU [37]	**CReLU [32]**	**zReLU [31]**	**2SReLU [14]**	**PhaseReLU**
Acc	90.550	89.049	84.931	89.926	90.843

One of the crucial processes of convolutional neural networks is down-sampling, which reduces the image resolution to prevent over-fitting and latency of the model. For example, the max-pooling layer and average-pooling layer are widely used. The most famous pooling method in the Fourier domain, presented as the substitute of max-pooling, is spectral pooling [13]. Spectral pooling is explained as cropping the low-frequency components, the center of the image, without losing any considerable image information values. However, because we omit the shift operation, the image is truncated at four edges. In order to implement our down-sampling the image, the cyclic shifting theorem introduced in the FFTW library [38] is considered. Our proposed sub-sampling spectral pooling is represented in two steps. First, the kernel is padded and expanded with the input size. Second, the kernel is shifted and wraps around the image in two dimensions considering the kernel cases. According to Figure 5, when the activation map is denoted as I and the expected size of the kernel is $k \times k$, the spectral non-shift pooling method is described in two ways, depending on whether the kernel size is divisible by four. For example, given $k^{4\times4}$, which can be divided by four (8 mod 4 = 0), and $k^{6\times6}$, which cannot divided by four (6 mod 4 = 1), they are represented as following the top of Figure 5 and the bottom of Figure 5, respectively. Our proposed low-frequency cropping method without using the cyclic concept is shown on the right side of Figure 5.

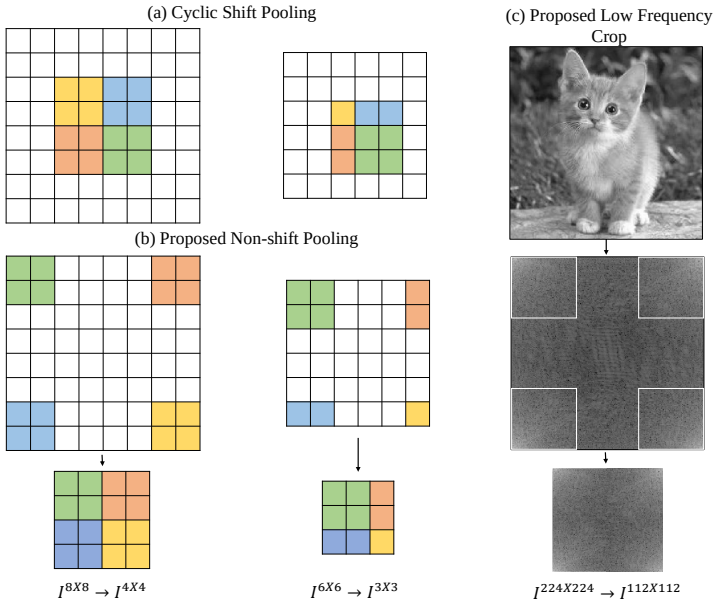


Figure 5. Comparison between standard cyclic shift pooling and our proposed non-shift pooling. (**a**) Cyclic pooling method. (**b**) Non-shift pooling method. The left and right sides of the figure for (**a**,**b**) describe a kernel divisible by 4 (e.g., size 8×8) and a kernel size not divisible by 4 (e.g., size 4×4). (**c**) Our proposed non-shift pooling method. The image is one example of a free download that is 224×224 in size and down-sampled into 112×112.

3.2.3. Architecture of Efficient Fourier CNN

As shown in Figure 6, let us assume that X_i, k_i, R_i and $\alpha_{\Re_i or \Im_i}$ represent the ith spatial input, spatial kernel, random kernel and scalar, respectively. Each $F(y)_i$ and $F(y')_i$ for ith layer refers to the output of a Fourier CNN and an efficient Fourier CNN, respectively.

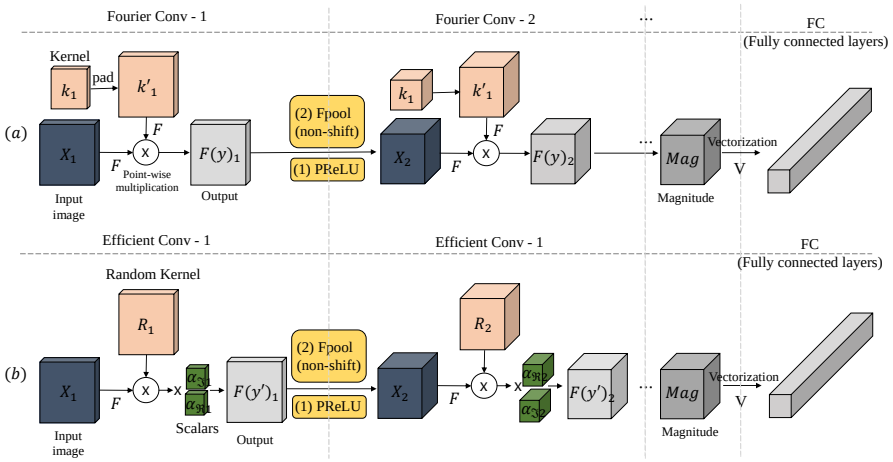


Figure 6. Architectures of presented convolutional neural networks (CNNs) for reducing the number of parameters. (**a**) Proposed Fourier CNN. (**b**) Proposed efficient Fourier CNN.

Both architectures of the Fourier CNN and efficient Fourier CNN are entirely trained in the frequency domain but slightly differ in applying the kernel. First, the kernel of the Fourier CNN is filled with zeros to fit the spatial domain to the same size as the input image size. Compared to conventional CNNs in the spatial domain, Fourier CNNs are set up with a 3×3 size kernel for a given CNN-based model, requiring only nine parameters per channel. Then, the Fourier transformed kernel is multiplied by the Fourier transformed image. However, there are concerns that the number of parameters of a Fourier CNN can increase dramatically as the number of layers deepens. Thus, we propose another method, an efficient Fourier CNN, based on a compressed random kernel that exponentially reduces the parameters. The bottom of Figure 6 shows that the efficient Fourier CNN starts from the multiplication of the random kernel generated by Gaussian distribution and the Fourier transformed image. For an output feature map of complex values, the scale of a Gaussian distribution in the complex domain is expected to be learned by multiplying the scalar or alpha values for real and imaginary numbers, respectively.

After the convolutional layer is performed, the Fourier-based activation function, PhaseReLU (or PReLU) is implemented. Since no cyclic shift theorem is applied to the pooling method in the frequency domain, the activation map is conducted with non-shift spectral pooling (FPool). Calculating the magnitude (*Mag*) containing important image information before reaching the fully connected layer (FC layer) converts the values to real numbers rather than complex numbers. Finally, the FC layer is implemented after vectorization to classify a given image.

To summarize, compared with prior studies on convolutional neural networks in the frequency domain, the proposed methods have three main differences. First, we focus on the light-weighting of the CNN-based model in the frequency domain. Second, we introduce a method to entirely implement CNNs in the Fourier domain without inverse Fourier transform. Third, we present a competitive activation function in the frequency domain.

4. Experimental Results

We evaluate our proposed methods, Fourier CNN (F-CNN) and efficient Fourier CNN (EF-CNN), in the frequency domain on various gray-scale image datasets for shallow neural networks and deep neural networks. We convert color image datasets, such as CIFAR-10, CIFAR-100 and SVHN, to gray to set all datasets as gray-scale images. We also

measure the accuracy versus the number of parameters for our two methods and a baseline method. The baseline method is a traditional convolutional neural network (CNN) in the spatial domain.

4.1. Analysis of Fourier CNN

We evaluate Fourier CNN for shallow networks, especially for LeNet-3 and LeNet-5. In order to reduce the number of parameters, we conduct an experiment using a 3×3 size kernel in the Fourier domain instead of the traditional 5×5 sized kernel in the spatial domain. All results of the model accuracy are the average of the last ten epochs over ten different files.

4.1.1. Shallow Fourier Neural Network

We first test the Fourier CNN method on a 28×28 size MNIST dataset consisting of 10 classes with 60,000 training images and 10,000 test images. We examine our method on the Fashion-MNIST dataset of size 28×28, which has the same number of training and validation sets as MNIST, consisting of 10 classes, but is more complex to learn.

To observe the results of training the presented method on larger gray-scale images, we evaluate 32×32 gray-scale image datasets, such as CIFAR-10 (G), CIFAR-100 (G), and SVHN(G). The CIFAR-based dataset and SVHN consist of 50,000 and 73,257 for training images and 10,000 and 26,032 for test images, respectively. There are three primary purposes for observing these datasets. First, since these are more complex than the original gray-scale images, we make sure that our method is well trained on complex images. Second, we check how our method is validated for more classes through the CIFAR-100 (G), which consists of 100 classes. Third, we examine the applicability of our proposed method for real-world data with SVHN, a real-world numerical data set.

We train Fourier CNN on LeNet-3 and LeNet-5 with one GPU at 70 epochs with a mini-batch size of 128. The model is optimized by stochastic gradient descent (SGD), using a double sigmoid function with the learning rates 0.001, 0.01, and 1×10^{-5} for the initial, top and final, respectively. We also adopt the weight decay of 0.0005 and the momentum of 0.9 for SGD. The weight parameters of kernels are initialized by a normal distribution with a mean of 0 and a std. of 1.

We compare the accuracy and the parameter quantities of existing CNNs, fully connected layer (FC) and our Fourier CNN (F-CNN), for LeNet-3 in Table 2 and LeNet-5 in Table 3 on given various datasets. To verify the performance of the convolutional layer of the CNN-based model, we also conduct an experiment on a simple fully connected layer to which the convolutional layer is not applied. According to Tables 2 and 3, we confirm that the existing CNN and F-CNN have higher accuracy than the FC-based method for both LeNet-3 and LeNet-5 on all datasets; therefore, we argue that the convolutional layer that extracts image features plays an important role in increasing the accuracy.

We further observe the accuracy per the number of parameters of CNN in the spatial domain and our method, F-CNN. Table 2 shows that our F-CNN on LeNet-3 mostly outperforms CNN and implements similar results in all datasets, despite using a kernel of size 3×3 to reduce 1654 parameters. In particular, we find that the accuracy of our F-CNN is improved by 0.1%, 0.4% and 1.5% on Fashion-MNIST, CIFAR-100(G) and SVHN, respectively. We have further experiments on LeNet-5, as shown in Table 3, where F-CNN also leads to competitive accuracy with CNN in most of the datasets. For example, each Fashion-MNIST, CIFAR-10(G), and CIFAR-100(G) improve accuracy by about 0.1%, 0.4% and 1.4%, respectively, and SVHN shows the same results as the existing method.

To summarize, our presented Fourier CNN shows higher or competitive results on an image classification task with a small number of weight parameters in the shallow network.

4.1.2. Deep Fourier Neural Networks: VGG09, VGG11 and VGG13

We conduct several experiments for deep neural networks, such as VGG-09, VGG-11 and VGG-13, on the same dataset conducted in the shallow networks. In order to prevent

over-fitting in the deep neural network and check the performance of our method, we resize the image resolution to double the size. For instance, MNIST and Fashion-MNIST are expanded to 56×56, and CIFAR-10, CIFAR-100, and SVHN are resized to 64×64. The hyperparameters and experimental environment settings are applied the same as for the shallow network, except for 100 epochs.

Table 2. Accuracy (%) of LeNet-3 on gray-scale image datasets.

Dataset	Method					
	CNN		FC		F-CNN	
	Acc.	Params	Acc.	Params	Acc.	Params
MNIST	98.8	107,786	96.8	66,790	98.6	106,132
FashionMNIST	90.7	107,786	87.4	66,790	90.8	106,132
CIFAR-10(G)	70.4	136,586	34.2	94,600	69.9	134,932
CIFAR-100(G)	66.9	144,236	34.1	94,600	67.3	142,582
SVHN(G)	82.8	136,586	72.3	86,950	84.3	134,932

Table 3. Accuracy (%) of LeNet-5 on gray-scale image datasets.

Dataset	Method					
	CNN		FC		F-CNN	
	Acc.	Params	Acc.	Params	Acc.	Params
MNIST	98.8	545,546	97.6	105,214	98.6	513,052
FashionMNIST	90.6	545,546	88.0	105,214	90.7	513,052
CIFAR-10(G)	62.0	696,746	40.8	134,014	62.4	664,252
CIFAR-100(G)	63.6	704,396	40.3	141,664	65.0	671,902
SVHN(G)	88.4	696,746	80.3	134,014	88.4	664,252

Table 4 shows the accuracy results of CNN and Fourier CNN for VGG-09 and VGG-11. Overall, in the VGG-09 model, F-CNN shows almost the same accuracy rate as traditional CNN, and in VGG-11, it is improved by 0.1% and 0.2% in FashionMNIST and SVHN, respectively. We further test on the VGG-13 model as shown in Table 5. As a result, our F-CNN has almost the same accuracy as CNN in the spatial domain for all datasets and shows an improvement of 0.9% on CIFAR-100. In conclusion, we argue that CNN in the spatial domain can be replaced with our proposed method, F-CNN, in the Fourier domain.

Table 4. Accuracy (%) of VGG-09 and VGG-11 on gray-scale image datasets.

	Model			
	VGG-09		VGG-11	
Dataset Method	CNN	F-CNN	CNN	F-CNN
MNIST	99.3	99.3	99.1	99.1
FashionMNIST	91.3	91.0	91.2	91.3
CIFAR-10(G)	77.4	77.0	79.1	79.1
CIFAR-100(G)	77.4	77.4	79.1	79.1
SVHN	92.8	92.8	92.6	92.8

4.2. Analysis of Efficient Fourier CNN

According to the previous observation, we attempt to reduce weight by converting the 5×5 kernel size to a 3×3 kernel size in the shallow network. For further work, we test our efficient Fourier CNN (EF-CNN) in shallow networks with even fewer parameters in order to check whether EF-CNN is competitive to conventional CNN. We train our method on a model with epoch 70, and we use the same hyper-parameters as in Section 4.1.1. In the

weight initialization, the imaginary part random filter and the real part random filter are initialized to the normal distribution, respectively, and the scalars for real and imaginary are initialized to 1.

Table 5. Accuracy (%) of VGG-13 on gray-scale image datasets.

Dataset	Method	
	CNN	**F-CNN**
MNIST	99.2	99.2
FashionMNIST	91.3	91.3
CIFAR-10(G)	81.0	81.0
CIFAR-100(G)	81.2	82.1
SVHN	93.4	93.4

4.2.1. Accuracy

Tables 6 and 7 show the results of the experiments on the CNN, FCNN, and EF-CNN models on gray-scale image datasets from LeNet-3 and LeNet-5, respectively. The advantage of the efficient Fourier CNN is verified to achieve very similar performance to CNN with fewer parameters in shallow networks.

Table 6. Efficient Fourier CNN accuracy (%) of LeNet-3 on gray-scale datasets.

Dataset	Method		
	CNN	**F-CNN**	**EF-CNN**
MNIST	98.8	98.6	98.6
FashionMNIST	90.7	90.8	90.5
CIFAR-10(G)	70.4	69.9	69.9
CIFAR-100(G)	66.9	67.3	66.8
SVHN	83.8	84.3	83.8

Table 7. Efficient Fourier CNN accuracy (%) of LeNet-5 on gray-scale datasets.

Dataset	Method		
	CNN	**F-CNN**	**EF-CNN**
MNIST	98.8	98.6	98.6
FashionMNIST	90.6	90.7	90.6
CIFAR-10(G)	62.0	62.4	62.0
CIFAR-100(G)	63.6	65.0	63.2
SVHN	88.4	88.4	88.4

4.2.2. The Measurement on a Change in the Number of Parameters

To summarize, one of our proposed methods, F-CNN, uses a 3×3 kernel size to perform image classification with a high accuracy rate, applying a small number of parameters in shallow networks. The other method is EF-CNN, which uses a random filter instead of the conventional Fourier kernel in order to learn real and imaginary scalars for each channel.

As shown in Table 8, we compare the number of parameters when training a 28×28 sized image dataset in LeNet-5. First, in the case of Fourier CNN, the number of trainable parameters in the first convolutional layer (Conv1), Conv2, and Conv3 are reduced by about $\times 0.34$, $\times 0.35$, and $\times 0.35$, respectively. In addition, the total trainable parameter in LeNet-5 is reduced to $\times 0.94$.

Table 8. The number of parameters of LeNet-5 on the Fashion-MNIST dataset for each CNN, Fourier CNN and efficient FCNN. **TP**: trainable parameters. **Non-TP**: non-trainable parameters.

Architecture Layer	Parameter Type	Method					
		CNN		F-CNN		EF-CNN	
		Params	Ratio	Params	Ratio	Params	Ratio
Conv1	TP(#)	156	×1	54	×0.34	12	×0.07
	Non-TP(#)	-	-	-	-	9408	-
Conv2	TP(#)	2416	×1	864	×0.35	192	×0.07
	Non-TP(#)	-	-	-	-	37,632	-
Conv3	TP(#)	48,120	×1	17,280	×0.35	3840	×0.07
	Non-TP(#)	-	-	-	-	188,160	-
FC1	TP(#)	494,004		494,004		494,004	
FC2	TP(#)	850		850		850	
	Total TP(#)	545,546	×1	513,052	×0.94	498,898	×0.91

According to our experiment on EF-CNN, all three layers are reduced to ×0.07. Therefore, we notice that a total of ×0.91 trainable parameters are trained for our method. The result of Table 8 shows that our proposed methods can dramatically reduce the number of weight parameters in convolutional layers. Since the convolutional layer deepens for state-of-the-art deep neural networks, which require a massive amount of trainable parameters, our proposed methods have the potential to efficiently train with fewer parameters. We also observe the comparison of the total number of trainable parameters for the shallow network, according to Table 9. Our final statement about the experiments of F-CNN and EF-CNN is that instead of traditional CNNs in spatial domains where more parameters are required, these two methods can become future replacements for training image classification tasks.

Table 9. The total number of trainable parameters for CNN, Fourier CNN and efficient Fourier CNN based on random kernel.

Dataset	Method	Model			
		LeNet-3		LeNet-5	
		Params	Ratio	Params	Ratio
MNIST	CNN	107,786	×1	545,546	×1
	F-CNN	106,132	×0.98	513,052	×0.94
	EF-CNN	105,418	×0.97	498,898	×0.91
Fashion-MNIST	CNN	107,786	×1	545,546	×1
	F-CNN	106,132	×0.98	513,052	×0.94
	EF-CNN	105,418	×0.97	498,898	×0.91
CIFAR-10	CNN	136,586	×1	696,746	×1
	F-CNN	134,932	×0.98	664,252	×0.95
	EF-CNN	134,218	×0.98	650,098	×0.93
CIFAR-100	CNN	144,236	×1	704,396	×1
	F-CNN	142,582	×0.98	671,902	×0.95
	EF-CNN	141,868	×0.98	657,748	×0.93
SVHN	CNN	136,586	×1	696,746	×1
	F-CNN	134,932	×0.98	664,252	×0.95
	EF-CNN	134,218	×0.98	650,098	×0.93

5. Discussion

In this work, we demonstrated that CNNs in the Fourier domain, using our presented activation function, PhaseReLU, and random kernel, have the potential for computational efficiency, especially in reducing the number of parameters.

One possible further task is to extend the image to the video task and consider different types of CNNs, such as LSTMs and RNNs. Other types of transforms, such as wavelets and DCT, also can be candidates. In addition, computational costs and resources need to be considered in future work, which can be addressed by importing custom libraries instead of using our manually coded system.

6. Conclusions

We have transformed the convolutional neural network in the spatial domain to the frequency domain by Fourier transformation to perform image classification in reducing parameters efficiently. We also have introduced the newly proposed PhaseReLU, which is equivalent to ReLU in the spatial domain. In addition, we have removed the unnecessary step of the shift theorem operation used in the Fourier domain, thereby opening the possibility of preventing latency in the future. In order to implement the CNN weight parameters reduction technique in the Fourier domain, the Fourier CNN was always trained with a kernel size of 3×3 in the Fourier domain, regardless of the kernel size in the spatial domain. It was proved through the experiment of shallow networks and deep layered networks, and it showed competitive performance. Furthermore, the efficient Fourier CNN was newly proposed by applying a random kernel, using the principle of compression sensing to Fourier CNN. In the previous experiment, we obtained an accuracy very similar to that of the existing shallow network in the spatial domain, even if the simple scalar was learned. In conclusion, we have proposed a method to train image classification with a small number of parameters through the convolutional neural network in the Fourier domain. Furthermore, the advantage of the proposed kernel method is that it can be applied to any architectures based on CNNs, and exponentially reduces the number of weighting parameters for the convolutional layer. Future research is expected to develop a method for learning well, even in deep neural networks, and improving the speed aspect by taking advantage of point-wise multiplication in the Fourier domain, which requires much less computation than convolution in the spatial domain.

Author Contributions: Conceptualization, Y.H. and B.-W.H.; methodology, Y.H. and B.-W.H.; software, Y.H.; validation, Y.H. and B.-W.H.; formal analysis, Y.H.; investigation, Y.H.; resources, B.-W.H.; data curation, Y.H.; writing—original draft preparation, Y.H.; writing—review and editing, Y.H.; visualization, Y.H.; supervision, B.-W.H.; project administration, B.-W.H.; funding acquisition, B.-W.H. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Research Foundation of Korea: NRF-2017R1A2B4006023, NRF-2019K1A3A1A77074958, NRF-2020K2A9A2A06026659 and by the Korea government (MSIT): IITP-2021-0-01341, Artificial Intelligence Graduate School, Chung-Ang University and IITP-2021-0-01574, High-Potential Individuals Global Training Program.

Data Availability Statement: The datasets presented in this study are openely available in official Pytorch website at https://pytorch.org/vision/stable/datasets.html.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Lecun, Y.; Bengio, Y. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
2. Brigham, E.O. *The Fast Fourier Transform and Its Applications*; Prentice-Hall: Hoboken, NJ, USA, 1998.
3. Mathieu, M.; Henaff, M.; LeCun, Y. Fast training of convolutional networks through ffts. *arXiv* **2013**, arXiv:1312.5851.
4. Heckbert, P. Fourier transforms and the fast fourier transform (fft) algorithm. *Comput. Graph.* **1995**, *2*, 15–463.

5. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
6. He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.-J.; Han, S. Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
7. Zhang, X.; Zhou, M.; Lin, J.S. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
8. Yang, T.-J.; Howard, A.; Chen, B.; Zhang, X.; Go, A.; Sze, V.; Adam, H. Netadapt: Platform-aware neural network adaptation for mobile applications. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
9. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
10. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
11. Theodoropoulos, P.; Spandonidis, C.C.; Giordamlis, C.; Fassois, S. Stability and Safety of Ships and Ocean Vehicles Enhancing vessel's operational safety using Convolutional Neural Networks and Big Data techniques. In Proceedings of the 1st International Conference on the Stability and Safety of Ships and Ocean Vehicles (STAB&S), Online, 6–11 June 2021.
12. Chen, L.; Zhenya, W.; Bo, Z. Intelligent fault diagnosis of rolling bearing using hierarchical convolutional network- based health state classification. *Adv. Eng. Inform.* **2017**, *32*, 139–151.
13. Rippel, O.; Snoek, J.; Adams, R.P. Spectral representations for convolutional neural networks. *arXiv* **2015**, arXiv:1506.03767.
14. Watanabe, T.; Wolf, D.F. Image classification in frequency domain with 2SReLU: A second harmonics superposition activation function. *arXiv* **2020**, arXiv:2006.10853.
15. Ayat, S.O.; Khalil-Hani, M.; Ab Rahman, A.A.H.; Abdellatef, H. Sayed Omid Spectral-based convolutional neural network without multiple spatial-frequency domain switchings. *Neurocomputing* **2019**, *364*, 152–167. [CrossRef]
16. Lin, J.; Ma, L.; Yao, Y. A Fourier domain acceleration framework for convolutional neural networks. *Neurocomputing* **2019**, *364*, 254–268. [CrossRef]
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 770–778.
18. Iandola, F.; Moskewicz, M.; Karayev, S.; Girshick, R.; Darrell, T.; Keutzer, K. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv* **2014**, arXiv:1404.1869.
19. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
20. Liu, Y.; Lu, F. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105.
21. Anwar, S.; Sung, W. Compact deep convolutional neural networks with coarse pruning. *arXiv* **2016**, arXiv:1610.09639.
22. Hu, H.; Peng, R.; Tai, Y.W.; Tang, C.K. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv* **2016**, arXiv:1607.03250.
23. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
24. Han, S.; Huizi M.; William J.D. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
25. Pratt; Harry. Fcnn: Fourier convolutional neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Springer: Cham, Switzerland, 2017.
26. Franzen, F.; Yuan, C. Visualizing image classification in fourier domain. In Proceedings of the ESANN, Bruges, Belgium, 24–26 April 2019.
27. Wang, Z. Combining FFT and spectral-pooling for efficient convolution neural network model. In Proceedings of the 2016 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016), Beijjing, China, 20–21 November 2016; pp. 203–206.
28. Ryu, J.; Yang, M.S.; Lim, J. Dft-based transformation invariant pooling layer for visual classification. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
29. Bracewell, R.N.; Bracewell, R.N. *The Fourier Transform and Its Applications*; McGraw-Hill: New York, NY, USA, 1986; Volume 11, pp. 267–272.
30. Arjovsky, M.; Shah, A.; Bengio, Y. Unitary evolution recurrent neural networks. In Proceedings of the International Conference on Machine Learning (ICML), New York, NY, USA, 19–24 June 2016; pp. 1120–1128.
31. Nitzan, G. On complex valued convolutional neural networks. *arXiv* **2016**, arXiv:1602.09046.
32. Trabelsi, C.; Bilaniuk, O. Deep complex networks. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
33. Guan, B.; Zhang, J.; Sethares, W.A.; Kijowski, R.; Liu, F. Spectral Domain Convolutional Neural Network. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 2795–2799.
34. Donoho, D.L. Compressive sensing: From theory to applications, a survey. *J. Commun. Netw.* **2013**, *15*, 443–456.

35. Chen, Z.; Dongarra, J. Condition numbers of Gaussian random matrices. *SIAM J. Matrix Anal. Appl.* **2006**, *27*, 603–620. [CrossRef]
36. Ko, J.H.; Mudassar, B.; Na, T.; Mukhopadhyay, S. Design of an energy-efficient accelerator for training of convolutional neural networks using frequency-domain computation. In Proceedings of the 54th ACM/EDAC/IEEE Design Automation Conference (DAC) IEEE, Austin, TX, USA, 18–22 June 2017.
37. Nair, V.; Geoffrey, E.H. Rectified linear units improve restricted boltzmann machines. In Proceedings of the International Conference on Machine Learning (ICML), Haifa, Israel, 21–24 June 2010.
38. Podlozhnyuk, V. FFT-Based 2D Convolution. *NVIDIA White Paper*; 2007; Volume 32. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.472.2396&rep=rep1&type=pdf (accessed on 6 June 2021).