IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 September 2022, accepted 27 September 2022, date of publication 28 September 2022, date of current version 7 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3210710

RESEARCH ARTICLE

Principal Components of Neural Convolution Filters

SHOTA FUKUZAKI AND MASAAKI IKEHARA, (Senior Member, IEEE)
Department of Electronics and Electrical Engineering, Keio University, Yokohama-shi, Kanagawa 223-8522, Japan
Corresponding author: Shota Fukuzaki (fukuzaki@tkhm.elec.keio.ac.jp)

ABSTRACT Convolutions in neural networks are still essential on various vision tasks. To develop neural convolutions, this study focuses on Structured Receptive Field (SRF), representing a convolution filter as a linear combination of widely acting designed components. Although SRF can represent convolution filters with fewer components than the number of filter bins, N-Jet, the sole component system implementation, requires ten trainable parameters per filter to improve accuracy even for 3×3 convolutions. Hence, we aim to formulate a new component system for SRF that can represent valid filters with fewer components. Our component system named "OtX" is based on the Principal Component Analysis of well-trained filter weights because the extracted components will also be principal for neural convolution filters. In addition to proposing the component system, we develop a component scaling method to defuse massive scale differences among the coefficients in a linear combination of OtX components. In the experimental section, we train image classification models on CIFAR-100 dataset under the hyperparameters tuned for the original models with the standard convolutions. For NFNet-F0 classifier, OtX with six components performs 0.5% better than the standard convolution, 3.1% better than N-Jet with six components, and only 0.1% worse than N-Jet with ten components. Besides, OtX with nine components provides stabler training than N-Jet, performing 0.5% better than the standard for NFNet-F0. OtX suits when replacing standard convolutions because OtX performs at least comparably against N-Jet with further parameter efficiency and training stability.

INDEX TERMS Convoluton layer, Hermitian polynomials, neural network, structured receptive field.

I. INTRODUCTION

Neural Networks (NN) are essential in vision tasks because of their outstanding performance. Convolutional Neural Networks (CNNs) are common forms mainly consisting of convolution layers. CNNs sequentially convolve feature maps and extract suitable features for each task. The versatility of CNNs has made them predominant in the image processing field. Recent studies actively develop non-CNN architectures because convolution is inadequate in merging two related but distant features in a calculation. Vision Transformer (ViT) [1] and its derivates consist of multi-head self attentions (MSAs) [2] and multi-layer perceptrons (MLPs). MSAs directly compare features while ignoring their distance and determine the value based on the comparison result. MLP-Mixer [3] and its derivates adopt spatial MLPs instead of MSAs to capture long-range dependencies. Although this trend may seem to ostracize convolution layers from NNs, their importance is being re-acknowledged. Convolution, a simple aggregation of local features, performs better for extracting features than MSA, especially from less processed maps [4]. Besides, though convolution layers seem to not be used in ViT derivates, major patch aggregations are equivalent to the processes in convolution layers where the kernel size and the stride are equal to the patch size. Convolution layer classes are used in programming codes, excluding when adopting the standardization and the affine transform for each patch. Thus, even though the current trend focuses on capturing long-range dependencies, the convolution layer is an essential component in NNs for vision tasks.

Designing convolution filters is a way of developing convolution layers in neural networks. In this paper, the word

The associate editor coordinating the review of this manuscript and approving it for publication was Chaker Larabi.

"filter" means a two-dimensional convolution filter composed of weights for signals on every offset in its receptive field. A filter affects a channel in the input three-dimensional feature map and maps to a channel of the output feature map. Convolution filter design (CFD) aims to develop convolution filters in hopes of improving the quality or reducing the trainable parameters by modifying the structure of a filter from an array of trainable parameters. This study is a CFD work.

Let us first define variables for designing filters. A convolution layer convolves an input feature map $X \in \mathbb{R}^{C_{\text{in}} \times D_1 \times D_2}$ with the height D_1, the width D_2, and the number of channels C_{in}. Assuming that downsampling with strides larger than one is done after the convolution as needed, the output of the convolution Y has $C_{\text{out}} \times D_1 \times D_2$ bins. Although the numbers of channels C_{in} and C_{out} are fixed for each convolution layer, the spatial resolutions D_1 and D_2 are arbitrary. Then, the filter weights Θ consist of $C_{\text{out}} C_{\text{in}}$ filters with a particular kernel size of $K_1 \times K_2$. With these variables, a convolution is described as:

$$Y_{c_{\text{out}},h,w} = \sum_{\forall c_{\text{in}}, \forall i, \forall j} \Theta_{c_{\text{out}},c_{\text{in}},i,j} X_{c_{\text{in}},h+\Delta h_i, w+\Delta w_j}, \tag{1}$$

where c_{out} and c_{in} are the indices for the output/input channels, h and w are the vertical/horizontal positions of the maps, and i and j are the vertical/horizontal positions of the filter. For each pair $(c_{\text{out}}, c_{\text{in}})$, a convolution is a weighted summation of $X_{c_{\text{in}}}$ based on the importance of the position $(\Delta h_i, \ \Delta w_j)$ away from (h, w). Note that the convolution is calculated under a value such as zero for offsets out of X. In this paper, each two-dimentional filter $\Theta_{c_{\text{out}},c_{\text{in}}}$ is regarded as a linear combination of Λ two-dimentional filter components $\{F_\lambda\}_{\lambda=1,2,\cdots,\Lambda}$:

$$\Theta_{c_{\text{out}},c_{\text{in}},i,j} = \sum_{\forall \lambda} \theta_{c_{\text{out}},c_{\text{in}},\lambda} F_{\lambda,i,j}, \tag{2}$$

where $\theta \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times \Lambda}$ is the array of coefficients for the filter components $\{F_\lambda\}$. This representation is seen in [5]. The coefficients correspond to (sometimes scaled) the trainable weight parameters of a convolution layer. In the case that a filter is represented as an array of $K_1 \times K_2$ parameters, each filter component F_λ is described as:

$$F_{\lambda,i,j} = \delta_{i,i_\lambda} \delta_{j,j_\lambda}, \tag{3}$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta.

Conventional convolution layers implicitly assume the standard bases for each filter, and the coefficients for the standard bases are optimized through training. This primitive approach makes every $K_1 \times K_2$ filter representable. However, the effective area of each parameter is strictly restricted. This restriction decreases the worth of every parameter, making it highly dependent on its offset. On the contrary, Structured Receptive Field (SRF) [5] employs filter components where each can act as a meaningful filter. Introducing SRF turns the role of each parameter in convolution layers from the importance of the offset to the importance of its corresponding component. When a filter is a Gaussian filter, for example, the conventional components require cooperatively tuned parameters. In contrast, only one parameter is required when a system of components contains the Gaussian filter component. This example implies that a well-designed system will reduce non-zero parameters, making the optimization simpler and easier. Thus, how to construct Λ filter components is essential, and this is the main topic of this study.

N-Jet [6] is introduced as an SRF component system in [5] and is the sole component system formulation for SRF. Namely, all SRF applications have been implemented with N-Jet components. N-Jet components act as local differential operators of which the orders are m for the vertical axis and n for the horizontal axis. With the local differential filter components, N-Jet approximates local signals with the Taylor expansion and weights each polynomial signal components. Although extracting features from the Taylor expansion is a versatile approach in Natural science, N-Jet still have two problems as a system of SRF components. The first problem is SRF on N-Jet performs worse than the conventional convolution when training on sufficiently large-scale datasets such as the ImageNet dataset [7]. This problem critically, which is not actually treated in this paper, shows there is a room of investigating another component system for SRF. The second problem is N-Jet components are highly correlated one another. Although this nature guarantees N-Jet practically requires at most fifteen components, the nature also detracts the efficiency of each component. For example, N-Jet on SRF with six components cannot outperform the conventional convolution. Composing an SRF with more efficient and less correlated components will make training easier and improve performance with less components.

This paper proposes a new SRF component system, "OtX,"[1] as the second formulation of SRF. OtX is formulated by modeling implicit principal components of well-trained neural convolution filters. Thus, OtX has orthogonality, in other words, the components of OtX are not correlated one another. We also define the rule for the ordering of OtX components, which contributes to picking a finite number of efficient components to train. Since OtX reveals efficient components to characterize neural convolution filters, SRF on OtX requires fewer components than on N-Jet to outperform the standard representation. OtX is also formulated based on the Hermite polynomials and Gaussian function similarly to N-Jet. The main change is the employment of radial symmetric and $\frac{\pi}{4}$ rotated line-symmetric components. This change accepts inseparable components in contrast to N-Jet components which are all separable. In addition, we propose a component scaling to make training of SRF on OtX easier. Whereas a single use of the two does not perform well, the combined use of OtX and the component scaling generates a

[1]The name "OtX" is only a sequence of symbol characters indicating three symmetry types. The character "O" symbolizes radial symmetry, "t" symbolizes line symmetry $\left(\varphi = 0, \frac{\pi}{2}\right)$, and "X" symbolizes line symmetry $\left(\varphi = \frac{\pi}{4}\right)$, where φ is introduced in Section IV.

synergistic effect. The component scaling makes OtX comparable with N-Jet as an SRF component system.

Summarizing the description above, the contributions of this paper are as below.

- We analyze the implicit principal components of neural convolution filters and reasonably formulate them. In the formulation, we also generalize the rule for the ordering of components.
- We apply the formulated OtX system as a new candidate for SRF components, and OtX provides a denser filter representation than N-Jet. Namely, training with OtX can obtain better filters with fewer component candidates than with N-Jet.
- We propose a component scaling method that helps the training of OtX filters.

II. RELATED WORKS

This work aims to improve neural convolution layers. In this section, we introduce related works that have phrasally similar purposes. Not all related works conflict with this study, and our proposals can be applied to some of these works. Works that can be combined with OtX are introduced in the latter part of this section.

Neural convolution filter development that conflict with this work is categorized into two types. Works of the first type align trainable parameters symmetrically. A parameter for a convolution filter affects multiple offsets in the filter. Applying these components reduces the number of trainable parameters. A work [8] develops horizontally symmetric alignments of parameters. SymKer [9] replicates a unit of parameters to form a filter. SymNet [10] uses three types of radial symmetric filters. Filters in these works are not spatially biased. However, the available shapes of filters in these works are too restricted to maintain the performance of neural networks. Hence, the works conclude that their proposals can reduce trainable parameters without salient performance degradation. Besides, these works cannot keep the number of parameters small for large receptive fields because the number scales linearly with K_1K_2. On the contrary, works of the second type succeed in small-scale experiments, and the number of parameters is independent of receptive field sizes. These works design filter components of the linear combination representation and train the coefficients. N-Jet [6] is introduced as the first implementation of component system (originally called "bases") for Structured Receptive Field [5]. N-Jet components are tensor dot products of the vertical and the horizontal Gaussian derivatives. Precisely, a continuous N-Jet component $J_{m,n}$ of which the order for x is m, and that for y is n is defined as:

$$J_{m,n}(x, y) = \frac{\partial^{m+n}}{\partial x^m \partial y^n} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \tag{4}$$

where x and y denote the offsets for the horizontal and the vertical directions, respectively. A Gaussian derivative is correlated with another which has the same order parity. Thus, available pairs of Gaussian derivatives for an N-Jet component are limited to ones such that the total orders are up to 4. N-Jet components are similar to $\{\Psi^{(0)}_{m,n}\}$ in OtX, but OtXs' are not correlated with one another because of their orthogonality. SRF can choose an upper limit of the total order of two Gaussian derivatives, and the number of parameters for a filter can be 1, 2, 6, 10, or 15. This limitation causes difficulty in reducing the total number of trainable parameters for 3 × 3 convolutions which are the most widely used. To reduce parameters fewer than nine, OtX can keep at most eight components, whereas N-Jet must reduce components to six. N-Jet on SRF outperforms standard convolution filters on smaller scales than ImageNet classification with 1000 image classes. FracSRF [11] forms a filter as a tensor dot product of two approximated fractional derivatives of Gaussian. FracSRF needs three parameters per filter, derivative orders for the vertical and the horizontal directions and the scale of the tensor dot product. FracSRF performs comparably against SRF despite fewer parameters. Because all filters in FracSRF are separable, applying OtX to FracSRF is difficult.

Not all N-Jet applications conflict with OtX. N-Jet Net [12] is an expansion of SRF such that the receptive field scales are changed, corresponding to the scale parameter of Gaussian. The nature that SRF components have non-zero values in a particular area of a receptive field enables this expansion. Since OtX components also have this nature, OtX can be expanded similarly to N-Jet Net. For the same reason, OtX can apply to SESN [13], which uses a filter on multiple scales. Note that components in SESN are implemented based on $\{\psi_n\}$ which are not Gaussian derivatives.

Weight standardization [14] is a powerful development for convolution weights. Weight standardization adjusts the mean and the variance of weight values for every output channel to be 0 and 1, respectively. Scaled weight standardization in [15] scales the standardized weights with trainable gain parameters defined for output channels of convolutions. Applying OtX to these causes a problem: subtracting the mean from the filters loses the role of the components. To avoid this, we slightly modify the way of standardizing weights. We divide weights by their second moment around 0 instead of their standard deviations without subtracting their means. This modified scaled weight standardization is used in our experiments' NFNet [16] implementation.

Deformable convolution [17], [18] has flexible offsets, reference positions for convolutions. The offsets are calculated by looking around each position. Then, standard convolution layers are used as the explorers. We can replace the standard convolution with OtX, which is a way of applying OtX to Deformable convolution.

III. ANALYZING IMPLICIT PRINCPAL COMPONENTS OF NEURAL CONVOLUTION FILTERS

In this section, we analyze convolution filter weights in a well-trained CNN and show our OtX design policy. The CNN model used for analysis is a VGG16 classifier [19] trained on the ImageNet dataset [7]. We use the weights data from a model zoo, PyTorch Image Models [20].

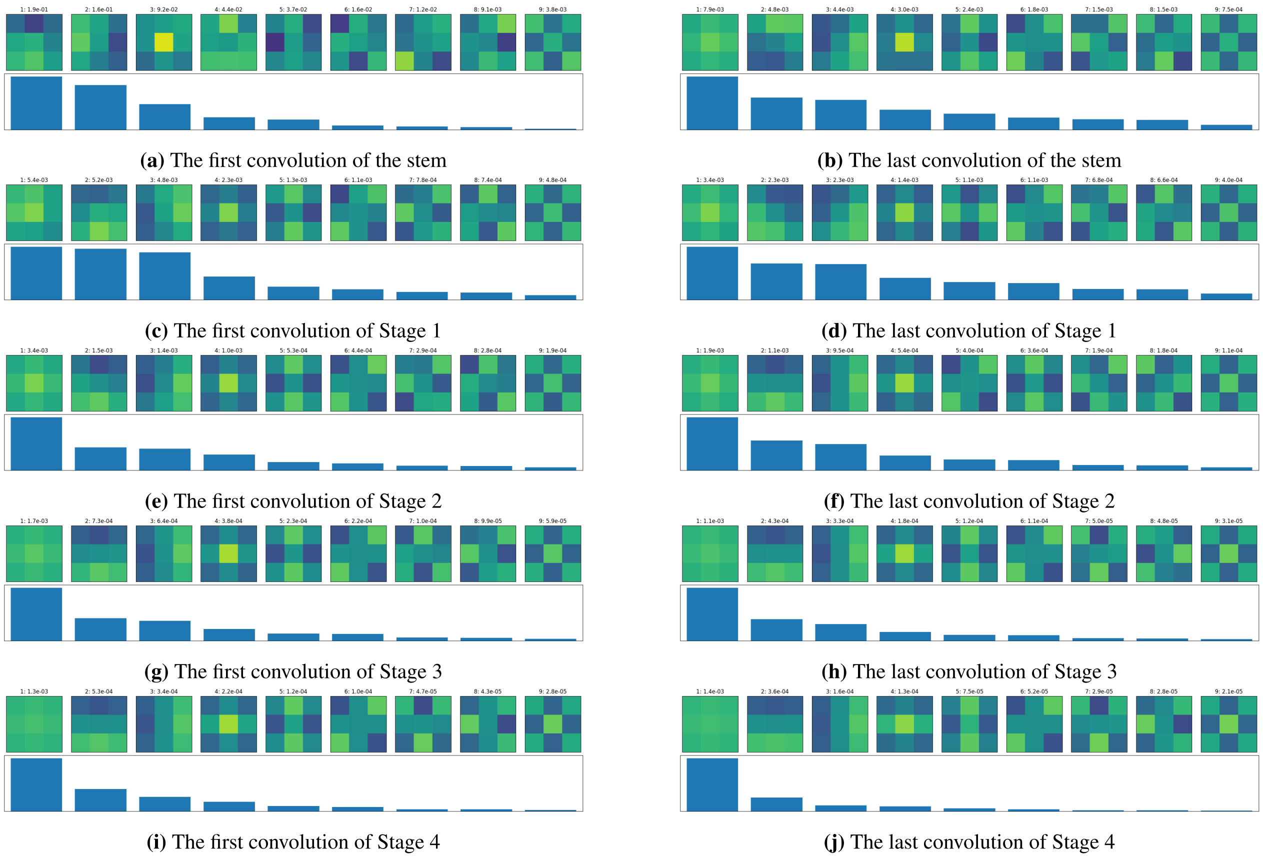

(a) The first convolution of the stem **(b)** The last convolution of the stem **(c)** The first convolution of Stage 1 **(d)** The last convolution of Stage 1 **(e)** The first convolution of Stage 2 **(f)** The last convolution of Stage 2 **(g)** The first convolution of Stage 3 **(h)** The last convolution of Stage 3 **(i)** The first convolution of Stage 4 **(j)** The last convolution of Stage 4

FIGURE 1. Results of a principal component analysis of VGG16 convolution filters. The components are sorted by their standard deviations indicated by the blue bar graphs.

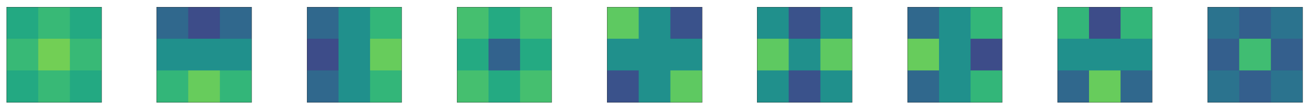
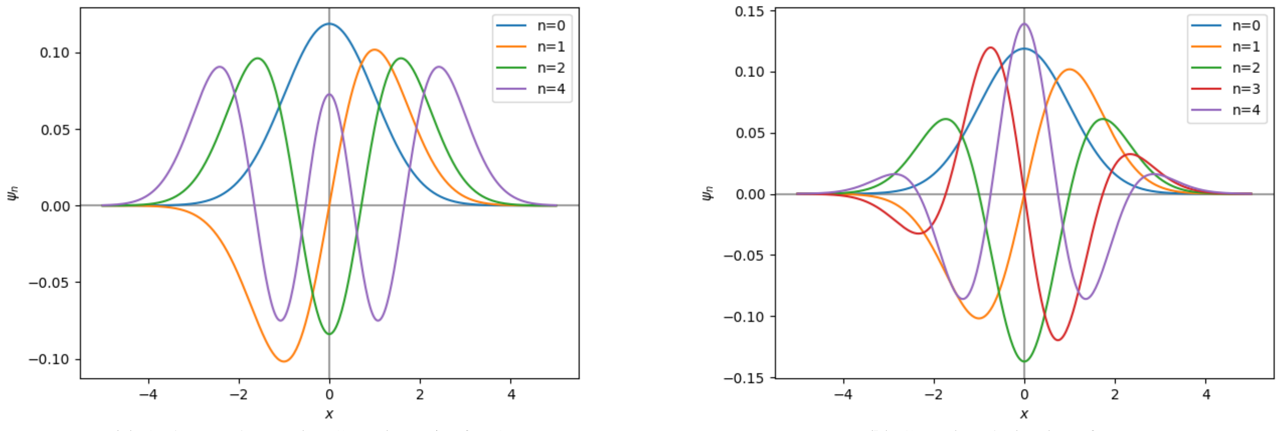

(a) Φ_0 **(b)** $\Psi_{0,1}^{(0)}$ **(c)** $\Psi_{0,1}^{(\frac{\pi}{2})}$ **(d)** Φ_2 **(e)** $\Psi_{1,1}^{(0)}$ **(f)** $\Psi_{1,1}^{(\frac{\pi}{4})}$ **(g)** $\Psi_{1,2}^{(0)}$ **(h)** $\Psi_{1,2}^{(\frac{\pi}{2})}$ **(i)** Φ_4

FIGURE 2. The most critical nine OtX components (3 × 3).

We show the result of Principal Component Analysis (PCA) [21], [22] of filter weights for each convolution layer in Figure 1. At a first glance, the principal components of each layer have almost a common characteristic. PCA is a way of extracting orthonormal bases from a set of vectors, and the bases are efficient for low-rank approximation of the vectors in the set. Note that the sign of each component is ignorable because the component is used for a linear combination. Bar graphs in Figure 1 visualize the standard deviations along with the components, and components with larger standard deviations can approximate with more minor errors. Namely, more left components in the figure are more critical for characterizing the filter weights of the convolution layers. This order is almost the same among all the layers, and it may imply the existence of a general efficient design of components for characterizing convolution filters. Formulating and applying these components will bring some benefits. For example, optimizations with carefully selected components will reduce the number of trainable parameters without significant deterioration. Thus, the remaining part of this section analyzes the PCA results and designs the OtX formulation policy.

(1) Every component is symmetric around its center point or two lines through its center. (2) Absolute values of outer points in a component tend to be smaller than that of inner points except for symmetrical axes of the component, which have odd symmetry. This result follows an intuition that signals in closer positions are more critical since convolution is a way of aggregating neighboring information. The decay rates of weights become smaller as the layer becomes far from the input. Thus, this decay rate should be adjustable for the layers. (3) Some components adjoin the rotation of



(a) Orthogonal Hermite-Gaussians ψ_n for OtX **(b)** Gaussian derivatives for N-Jet

FIGURE 3. **Two types of one-dimensional Hermite-Gaussians. The functions are numerically normalized so that their L_2 norms equal 1.**

themselves. There exist two types of angle differences: $\frac{\pi}{2}$ and $\frac{\pi}{4}$. Components with odd symmetry along either the horizontal or the vertical direction are the former type, and pairs of components where one of them has odd symmetry in both the horizontal and the vertical directions are the latter type. The latter component has almost the same odd symmetry in two directions. On the contrary, some components have no pair and accord to their $\frac{\pi}{2}$ rotation. These kinds of components can be regarded as radial symmetric components, which are isotropic, and each weight is determined by the distance from the center of the component. (4) The standard deviations of components seem to form a Gaussian distribution. Closer layers to the input have slower decay about the standard deviations. Notably, the ninth components at most have approximately $\frac{1}{10}$ standard deviations than the first components, and their contributions will be slight.

Let us summarize the above discussions. Each component is orthogonal to each other, and farther positions from its center have small absolute values. The components are categorized into two types, radially symmetric or line-symmetric, and each of the latter type filters has its pair which accord to itself when rotated. A line-symmetric component has at least one odd symmetric direction, and the rotation angle for the overlap is $\frac{\pi}{2}$ if it has only one or $\frac{\pi}{4}$ if it has two. The two symmetries are the same for a line-symmetric component with two odd symmetries. In the next section, we concretely formulate filter components that have these natures.

IV. FORMULATION

In this section, we formulate filter components and their priorities. First, we introduce one-dimensional orthogonal Hermite-Gaussians $\{\psi_n\}$ and note its basic natures. Second, we extend $\{\psi_n\}$ into two-dimensional functions in two ways. Our filter components are composed of these two types of functions. The first extension composes radial symmetric functions $\{\Phi_n\}$. These functions are defined if and only if n is even and are generally not separable. The second extension composes line-symmetric functions $\{\Psi^{(\varphi)}_{m,n}\}$. The details of the available (m, n, φ) combinations are described in Section IV-B2. After these extensions, we propose a way of giving priority scores for $\{\Phi_n\}$ and $\{\Psi^{(\varphi)}_{m,n}\}$. Introducing these scores enables one to choose Λ sufficiently effective components from infinite numbers of components. At last, we note a way of making $K_1 \times K_2$ numerical arrays from the continuous functions $\{\Phi_n\}$ and $\{\Psi^{(\varphi)}_{m,n}\}$.

A. ONE-DIMENSIONAL ORTHOGONAL HERMITE-GAUSSIANS

In this paper, we use "*physicist's* Hermite polynomials." Let n be a non-negative integer. Then, the n-degree Hermite polynomial $H_n(x)$ is defined by a recurrence relation

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \tag{5}$$

where $H_0(x) = 1$ and $H_1(x) = 2x$. Hermite polynomials $\{H_n(x)\}$ have orthogonality with a weight function $\exp(-x^2)$, namely

$$\int_{\mathbb{R}} H_m(x)\, H_n(x)\, \exp(-x^2)\, dx = \sqrt{\pi} 2^n n!\, \delta_{m,n}. \tag{6}$$

Thus, if an n-degree one-dimensional Hermite-Gaussian

$$\psi_n(x) = H_n(x)\, \exp\left(-\frac{x^2}{2}\right), \tag{7}$$

$\{\psi_n\}$ is an orthogonal system of functions, we have

$$\int_{\mathbb{R}} \psi_m(x)\, \psi_n(x)\, dx = \sqrt{\pi} 2^n n!\, \delta_{m,n}. \tag{8}$$

Note that N-Jet components [5] are constructed based on Gaussian derivatives $\left\{\frac{\partial^n}{\partial x^n} \exp\left(-\frac{x^2}{2}\right)\right\} = \left\{H_n\left(\frac{x}{\sqrt{2}}\right) \exp\left(-\frac{x^2}{2}\right)\right\}$, and the function system does not have orthogonality. Figure 3 shows graphs of normalized $\psi_n(x)$ and Gaussian derivatives for comparison.

If n is even, ψ_n is an even function, and if n is odd, ψ_n is an odd function. Therefore, $\psi_n(x)$ is either symmetric or anti-symmetric about $x = 0$. This symmetric nature guarantees that filter components' weights are aligned symmetrically.

B. ORTHOGONAL SYSTEM OF HERMITIAN FILTER COMPONENTS

1) EVEN DEGREE RADIAL SYMMETRIC FILTER COMPONENTS

We introduce a two-dimensional filter component system $\{\Phi_n\}$ by rotating ψ_n around the origin of an $\mathbb{R}^2$ plane. Φ_n is defined if and only if n is even because each Φ_n must be the same when rotating it by π. We assume an x-y plane such that its origin is the center of filter components, and the shorter side of the components corresponds to $[-1, 1]$. Then, let

$$\Phi_n(x, y) = \psi_n\left(\frac{\sqrt{x^2+y^2}}{\sigma}\right), \tag{9}$$

where σ is a trainable parameter that represents the spatial scale of components. Each convolution layer has one σ. For larger σ, filter components are bounded by the filter boundary, and for enough small σ, the components can have sufficient non-zero parts. Components shown as (a), (d), and (i) in Figure 2 are instances of Φ_n. Note that a Φ_0 is a Gaussian filter component. If $n \geq 2$, each Φ_n cannot be represented as $\Psi^{(\varphi)}_{m,n}$, because Φ_n is not separable for $n \geq 2$. In addition, $\{\Phi_n\}$ is an orthogonal system of $L_2(\mathbb{R}^2)$ as straightforwardly derived from the orthogonality of $\{\psi_n\}$. Therefore, each Φ_n can be a component obtained from a PCA if we ignore region truncation errors.

2) LINE SYMMETRIC FILTERS

We assume the same x-y space as Section IV-B1. Let a line-symmetric filter component

$$\Psi^{(\varphi)}_{m,n}(x, y) = \psi_m\left(\frac{x_\varphi}{\sigma}\right)\,\psi_n\left(\frac{y_\varphi}{\sigma}\right), \tag{10}$$

where (x_φ, y_φ) is a mapped point from the original position (x, y) by rotating it by φ around the origin of the x-y plane. Namely,

$$\begin{bmatrix} x_\varphi \\ y_\varphi \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}. \tag{11}$$

There also exist conditions for available (m, n, φ) combinations of $\{\Psi^{(\varphi)}_{m,n}\}$ as well as n of $\{\Phi_n\}$. Note that these are not limitations derived from the nature of ψ_n but conditions so that the components match the analysis result in Section III. Available (m, n, φ) combinations are elements of a set

$$\{(m, m, 0) \mid m \text{ is odd}\} \cup \left\{\left(m, m, \frac{\pi}{4}\right) \mid m \text{ is odd}\right\} \cup \{(m, m+1, 0)\} \cup \left\{\left(m, m+1, \frac{\pi}{2}\right)\right\}.$$

The available combinations of m and n are significantly limited to $n = m, m+1$, and only two φ, which equals 0 or another value, are defined for each relationship. The existence of Φ_n will prohibit the combination with even $m = n$. If $m = n$ is odd, the relationship $\Psi^{(\frac{\pi}{2})}_{m,m}(x, y) = \Psi^{(0)}_{m,m}(-x, y)$ holds, and that is the reason why $\varphi = \frac{\pi}{4}$ is used for odd $m = n$ combinations. Components shown as (b), (c), (e), (f), (g), and (h) in Figure 2 are instances of $\Psi^{(\varphi)}_{m,n}$. A system $\{\Psi^{(\varphi)}_{m,n}\}$ for available (m, n, φ) combinations is an orthogonal system, and each line-symmetric component in the system has orthogonality with all radial symmetric components. Therefore, elements of $\{\Psi^{(\varphi)}_{m,n}\}$ can be components of a PCA result simultaneously with $\{\Phi_n\}$ components if we ignore region truncation errors.

C. ORDER OF FILTER COMPONENTS

In Section IV-B, we defined two types of filter component systems $\{\Phi_n\}$ and $\{\Psi^{(\varphi)}_{m,n}\}$ as candidates for each F_λ in Eq. 2. In practice, we must pick finite Λ components from infinite candidates. Although this problem is generally to find the optimal combination, in this paper, we give a score for each component and greedily pick components that have smaller scores. We define the score as the total degree of one-dimensional orthogonal Hermite-Gaussians in the formulation of a component. The score is n for Φ_n and $m + n$ for $\Psi^{(\varphi)}_{m,n}$. Note that we prefer radial symmetric components over line-symmetric ones with the same scores in principle. There is an exception in the case of 2×2 filters where the order of the components is Φ_0, $\Psi^{(0)}_{0,1}$, $\Psi^{(\frac{\pi}{2})}_{0,1}$, $\Psi^{(0)}_{1,1}$ because the sampling results of Φ_2 equals that of Φ_0. This ordering with the score aligns with the component order in PCA as shown in Figure 2.

D. SAMPLING FROM THE CONTINUOUS FILTER COMPONENTS

Proposed filters are no longer arrays of trainable parameters, but they are still $K_1 \times K_2$ numerical arrays. Thus, continuous functions defined in Section IV-B must be sampled into $K_1 \times K_2$ arrays. In our definition of the x-y plane, the smaller side of a filter corresponds to $[-1, 1]$. We pick sample points $\{(x_j, y_i)\}$ for array indices $\{(i, j)\}$ by regular intervals on the x-y plane. Namely, we have

$$x_j = \frac{K_2 - 1}{K - 1}\left(\frac{2(j-1)}{K-1} - 1\right) \tag{12}$$

and

$$y_i = \frac{K_1 - 1}{K - 1}\left(\frac{2(i-1)}{K-1} - 1\right), \tag{13}$$

where $K = \mathrm{Min}\{K_1, K_2\}$. After calculating the filter component value for each (x_j, y_i), the array is divided by its L_2 norm. This normalization guarantees that the array norm equals a steady value of 1 even if the component is bounded.

V. COMPONENT SCALING

The PCA result shows that components in a layer have quite different coefficient variances. Thus, we attempt to create such differences in the variances among components. While a coefficient is usually one trainable parameter, we represent a coefficient as a product of two kinds of trainable parameters.

TABLE 1. Classification results on CIFAR-100 dataset. The phrase "The loss exploded." indicates the training fails because the loss value becomes too large to converge.

Model	Components	Weight Std.	Learning Rate	#Params	Top-1 Acc.(%)	Top-5 Acc.(%)
VGG16	Standard	-	0.0125	134 M	72.7	91.6
VGG16	N-Jet-10	-	0.0125	54.5 M	The loss exploded.	
VGG16	OtX-9	-	0.0125	50.8 M	73.4	91.9
DenseNet121	Standard	-	0.25	7.06 M	76.6	93.3
DenseNet121	N-Jet-10	-	0.25	7.29 M	The loss exploded.	
DenseNet121	OtX-9	-	0.25	7.05 M	77.3	93.0
EfficienNetV2-S	Standard	-	0.0625	20.3 M	79.3	94.4
EfficienNetV2-S	N-Jet-10	-	0.0625	20.4 M	The loss exploded.	
EfficienNetV2-S	OtX-9	-	0.0625	20.3 M	The loss exploded.	
NFNet-F0	Standard	Original	0.1875	68.7 M	81.7	96.1
NFNet-F0	N-Jet-10	Modified	0.1875	70.7 M	82.3	96.3
NFNet-F0	OtX-9	Modified	0.1875	68.7 M	82.2	96.2

TABLE 2. Relationships between the number of components and classification accuracy.

Components	#Params	Top-1 Acc.(%)	Top-5 Acc.(%)
N-Jet-6	62.9 M	79.1	95.2
N-Jet-10	70.7 M	82.3	96.3
N-Jet-15	80.4 M	80.1	95.5
OtX-6	62.9 M	82.2	96.4
OtX-8	66.8 M	82.2	96.3
OtX-9	68.7 M	82.2	96.2

Specifically,

$$\theta_{c_{\text{out}},c_{\text{in}},\lambda} = \alpha_{c_{\text{out}},c_{\text{in}},\lambda}\,\beta_\lambda, \tag{14}$$

where $\alpha_{c_{\text{out}},c_{\text{in}},\lambda}$ is a trainable parameter almost compatible with $\theta_{c_{\text{out}},c_{\text{in}},\lambda}$ and β_λ is a trainable parameter defined for the λ-th component in a layer. The newly introduced parameter β_λ adjusts the scales of coefficients for its corresponding component. This modification acts as a restriction such that coefficients, especially for components that do not have much effect, do not become too large. The increase in the number of trainable parameters is Λ. Since the number of $\{\theta_{c_{\text{out}},c_{\text{in}},\lambda}\}$ in a layer is $C_{\text{out}}\,C_{\text{in}}\,\Lambda$, this increase is negligibly small. In the case of $C_{\text{out}} = 32$ and $C_{\text{in}} = 3$, although this is one of the worst examples, the increase rate is approximately only 1%. We initialize β_λ with the equation

$$\beta_\lambda = \exp\left(-\frac{s_\lambda^2}{2}\right), \tag{15}$$

where s_λ is the score of the λ-th component. Scores for N-jet components are defined similarly to OtX in our ablation study.

VI. EXPERIMENTS

In this section, we keep the model structure of neural classification models but replace the convolution weights and compare validation results. Getting a better result from a method indicates that the method can find better convolution weights. Learning rates for training are tuned for the original models that obtain their convolution weights as arrays of trainable parameters. We use hyperparameters tuned for the original even when training weights on other representations. All experiments are done on three random seeds, and we show the average value over the three trials.

A. SETTINGS

We use TIMM [20] implementations as baseline classifier models. For filter architectures except for the baseline, we initialize convolution filter parameters so that the convolution weights have the same variance as the baseline. We experiment with classifications on CIFAR-100 benchmark dataset [23]. Training images are resized to 192 × 192 pixels with distorted bounding box crops and taken random horizontal flips [19]. Validation images are center-cropped to 224 × 224 pixels after being resized to 256 × 256 pixels by bicubic interpolation. After the transformations, we standardize image values for each channel by the mean and the standard deviation of the channel over the dataset. Mini-batch size is set as 32, and we train the models for 90 epochs. The loss function is the cross-entropy loss with label smoothing [24], and the smoothing parameter is set to 0.1. We employ the standard stochastic gradient descent (SGD) optimizer with Nesterov's accelerated gradient method, of which the momentum is set as 0.9. Learning rates fluctuate under the cosine decay schedule after the linear warmup over 5 epochs. We apply a Sharpness Aware Minimization (SAM) [25] step once every 5 training steps.

B. COMPARISON AGAINST THE BASELINE AND N-JET

We compare the OtX architecture against the baseline and a modified N-Jet with trainable Gaussian scales. The initial filter scale σ is set to 1. We denote the modified N-Jet with Λ components as N-Jet-Λ, and a notation "OtX-Λ" is defined similarly for the OtX architecture. We choose $\Lambda = 10$ for N-Jet, which is the best Λ, and $\Lambda = 9$ for OtX, which uses almost the same number of trainable parameters as standard 3 × 3 filters.

TABLE 3. Relationships between the number of components and classification accuracy. Rows that start with bold text are the same settings as Table 1. The phrase "The loss exploded." indicates the training fails because the loss value becomes too large to converge.

Components	orthogonality	component scaling	Top-1 Acc.(%)	Top-5 Acc.(%)
N-Jet-10	✓	✓	81.7	96.2
N-Jet-10	✓		82.2	96.3
N-Jet-10		✓	The loss exploded.	
N-Jet-10			82.3	96.3
OtX-9	✓	✓	82.2	96.2
OtX-9	✓		The loss exploded.	
OtX-9		✓	82.1	96.3
OtX-9			81.2	95.9

We experiment on four classification models; VGG16 [19], DenseNet-121 [26], EfficientNetV2-S [27], and NFNet-F0 [16]. Table 1 shows the results. Models with OtX convolutions outperform those with standard convolutions on the datasets where the models were able to be trained. Additionally, training on OtX is more stable than on N-Jet when hyperparameters are tuned for standard convolutions. Accordingly, standard convolutions can be replaced with OtX ones without modifying the hyperparameters. Employing N-Jet requires further hyperparameter-tunings even if good hyperparameters for the model are known. Despite the stability, OtX performs at least comparably with N-Jet. Thus, OtX can be a candidate for an SRF component system to obtain better filters without tuning any hyperparameters.

C. EFFICIENCY OF COMPONENTS

We train NFNet-F0 on N-Jet and OtX with various Λ. This experiment intends to reveal the efficiency of components of the systems and the trade-off between the expansion of representable filter space and the optimization of filters by increasing the number of components. Table 2 shows the results. Whereas N-Jet-6 performs worse than N-Jet-10, OtX-6, which has the same number of components, performs almost as well as OtX-9 and N-Jet-10. This comparison indicates that OtX provides six more efficient components for neural convolutions than N-Jet. In the second sight for the trade-off, OtX-6 seems to have the best number of components, and what has a larger Λ obtains worse filters. However, OtX-9 has two advantages over OtX-6. The first advantage is the max accuracy. The max accuracy with OtX-9 is 82.7 %, and that with OtX-6 is 82.4 %. This indicates that exploration of coefficients for better filters sometimes succeeds even in the case with a larger Λ. The second advantage is the stability of the training. In the three trials we experimented, OtX-6 failed in training of one of the trials, while OtX-9 was trained stably in all three trials. Thus, OtX-9 is more suitable when one wants to obtain better filters through multiple trials or avoid training failures.

D. ABLATION STUDY

We do an ablation study to reveal what difference between N-Jet and OtX contributes to the performance. There are three main differences between N-Jet and OtX; orthogonality of components, employment of radially or rotated line-symmetric components, and application of the component scaling. We do experiments on all combinations to test the efficacy of each property. Table 3 shows the results. Although each of the three properties influences the results, finding a general rule is difficult since none of the three improves the accuracies regardless of the other two properties. For example, employing the component scaling works well on OtX experiments, but it does not on N-Jet. Especially, comparing the results of Row 4 and Row 8 indicates that only applying new types of symmetry to N-Jet decreases the accuracy, and combined use with other two features, orthogonality and the component scaling, compensates the accuracy (Raw 5). Thus, the combination of the three is inseparable and may be suitable for finding better filters.

VII. CONCLUSION

In this paper, we design a new SRF component system named "OtX" for better optimization of neural convolution filters. OtX is a modeled formulation of the implicit principal components of neural convolution filters based on a PCA result of well-trained convolution filters. The designed components are radial or line-symmetric and approximately orthogonal to one another. Training convolution filters with OtX can find more efficient weights than the standard filters, which represented as arrays of trainable parameters, without tuning the hyperparameters of most of the original standard models. In the case of using the original's hyperparameters, OtX is more stable than the existing N-Jet, but further stability is required. Since OtX components are the principal ones of neural convolution filters, OtX is more suitable for representing filters with fewer components than N-Jet. More OtX components provide a stabler training and a higher probability of finding better filters. We also propose a component scaling for OtX that improves OtX optimization to a degree comparable to a successful trained N-Jet. Thus, our proposed method can be a useful SRF component system and can be applied even when applying SRF on N-Jet is difficult.

REFERENCES

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16 × 16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017. [Online]. Available: https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[3] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, and M. Lucic, "MLP-Mixer: An all-MLP architecture for vision," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021. [Online]. Available: https://openreview.net/forum?id=EI2KOXKdnP

[4] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "CoAtNet: Marrying convolution and attention for all data sizes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 3965–3977.

[5] J.-H. Jacobsen, J. Van Gemert, Z. Lou, and A. W. M. Smeulders, "Structured receptive fields in CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2610–2619.

[6] L. Florack, B. Ter Haar Romeny, M. Viergever, and J. Koenderink, "The Gaussian scale-space paradigm and the multiscale local jet," *Int. J. Comput. Vis.*, vol. 18, no. 1, pp. 61–75, Apr. 1996.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[8] S. X. Hu, S. Zagoruyko, and N. Komodakis, "Exploring weight symmetry in deep neural networks," *Comput. Vis. Image Understand.*, vol. 187, Oct. 2019, Art. no. 102786.

[9] V. Dudar and V. Semenov, "Use of symmetric kernels for convolutional neural networks," in *Proc. Int. Conf. Data Sci. Intell. Anal. Inf.*, 2018, pp. 3–10.

[10] G. Dzhezyan and H. Cecotti, "Symmetrical filters in convolutional neural networks," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 7, pp. 2027–2039, Jul. 2021.

[11] N. Saldanha, S. L. Pintea, J. C. V. Gemert, and N. Tomen, "Frequency learning for structured CNN filters with Gaussian fractional derivatives," in *Proc. 32nd Brit. Mach. Vis. Conf.*, Nov. 2021. [Online]. Available: https://www.bmvc2021-virtualconference.com/programme/accepted-papers/

[12] S. L. Pintea, N. Tomen, S. F. Goes, M. Loog, and J. C. van Gemert, "Resolution learning in deep convolutional networks using scale-space theory," *IEEE Trans. Image Process.*, vol. 30, pp. 8342–8353, 2021.

[13] I. Sosnovik, M. Szmaja, and A. Smeulders, "Scale-equivariant steerable networks," in *Proc. Int. Conf. Learn. Represent.*, 2020. [Online]. Available: https://iclr.cc/virtual_2020/poster_HJgpugrKPS.html and https://openreview.net/forum?id=HJgpugrKPS

[14] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-batch training with batch-channel normalization and weight standardization," 2019, *arXiv:1903.10520*.

[15] A. Brock, S. De, and L. S. Smith, "Characterizing signal propagation to close the performance gap in unnormalized resnets," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=IX3Nnir2omJ

[16] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," 2021, *arXiv:2102.06171*.

[17] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 764–773.

[18] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets v2: More deformable, better results," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9308–9316.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[20] R. Wightman. (2019). *PyTorch Image Models*. [Online]. Available: https://github.com/rwightman/pytorch-image-models

[21] F. R. S. K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *London, Edinburgh, Dublin Philosoph. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, 1901.

[22] C. M. Bishop, *Pattern Recognition and Machine Learning*. Cham, Switzerland: Springer, 2006.

[23] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Tronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf and https://www.cs.toronto.edu/~kriz/cifar.html

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[25] D. Bahri, H. Mobahi, and Y. Tay, "Sharpness-aware minimization improves language model generalization," in *Proc. Int. Conf. Learn. Represent.*, 2022. [Online]. Available: https://openreview.net/forum?id=6Tm1mposlrM

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[27] M. Tan and Q. Le, "EfficientNetV2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10096–10106.

SHOTA FUKUZAKI received the B.E. and M.E. degrees in electrical engineering from Keio University, Yokohama-shi, Japan, in 2019 and 2021, respectively, where he is currently pursuing the Dr.Eng. degree under the supervision of Prof. M. Ikehara. His research interests include image classification and object detection.

MASAAKI IKEHARA (Senior Member, IEEE) received the B.E., M.E., and Dr.Eng. degrees in electrical engineering from Keio University, Yokohama-shi, Japan, in 1984, 1986, and 1989, respectively. He was an Appointed Lecturer at Nagasaki University, Nagasaki, Japan, from 1989 to 1992. In 1992, he joined at the Faculty of Engineering, Keio University. From 1996 to 1998, he was a Visiting Researcher at the University of Wisconsin, Madison, and Boston University, Boston, MA, USA. He is currently a Full Professor with the Department of Electronics and Electrical Engineering, Keio University. His research interests include multi-rate signal processing, wavelet image coding, and filter design problems.