ADAPTIVE KERNEL SELECTION FOR CONVOLUTIONAL NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Neural Networks (CNN) are used for various applications ranging from computer vision to natural language processing. A kernel, known as the matrix of weights, performs a convolution operation on input data. In general, the optimizer updates the weights of the kernel. Recent research suggests that applying a deterministic kernel after convolution operation can help a CNN to gain better generalization. However, how to compute the weights of the deterministic kernel is still a field of active research. In this work, we derived a lemma that shows of the representativeness of an adaptive deterministic kernel. We construct an adaptive deterministic kernel based on the Gaussian distribution of convoluted data. We generate many set of kernels by shifting weights to different positions of the initially created Gaussian kernel. We notice a pattern of weight distribution in deterministic kernels constructed from the Gaussian distribution of convoluted data. Using the derived lemma, it is possible to sort out a set of kernels (from many set of kernels) that tends to gain better performance in CNN for image classification task. The main object of this research work[1] is to identify these patterns and recommend the better set of kernels to gain performance.

1 INTRODUCTION

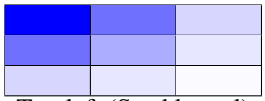
Convolutional Neural Network (CNN) gains impressive performance on computer vision tasks such as image classification (He et al., 2016), image segmentation (Long et al., 2015), and object detection (Redmon et al., 2016). A kernel (i.e., matrix of weights) performs convolution operations on input data to extract relevant features. The outcome of the convolution operation is the convoluted data, which is passed to the next layer. A CNN variant can have convolution, non-linear, and fully connected (FC) layers, and the order of these layers can vary based on the variant.

There are several ways to compute the weights of the kernel and distribute the weights into the kernel space (i.e., the size of the matrix). Saxe et al. (2011); Pinto et al. (2011); Jarrett et al. (2009); Salehinejad et al. (2022) gain better results by using randomly initialized kernels. Coates et al. (2011); Mairal et al. (2014); Cho & Saul (2009) use unsupervised criterion to learn the weights of the kernel. The usual practice is to use a single kernel in a CNN variant. Bo et al. (2010); Li et al. (2015) use multiple kernels and achieve better accuracy in CNN. In the context of image processing, the kernels are learnable parameters of the model. The weight of the kernel is updated via back-propagation by the optimizer based on the training loss/error.

Adaptive kernel shows an impressive performance in computer vision tasks (Du et al., 2017; Li et al., 2017; Xiong et al., 2015; Ding et al., 2018; Li et al., 2019). Recently, Curriculum by Smoothing (CBS) (Sinha et al., 2020) shows excellent results by building an adaptive kernel that controls the amount of high-frequency information propagated within the CNNs as training progress. CBS uses a Gaussian kernel after convolution

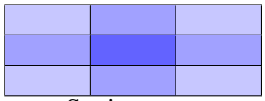
[1] https://github.com/PaperUnderReviewDeepLearning/KernelSet

.3315	.1889	.0554
.1889	.1076	.0315
.0554	.0315	.0092



Top-left (Seed kernel)

.0751	.1238	.0751
.1238	.2041	.1238
.0751	.1238	.0751



Static-center

Table 1: The weights of the Gaussian kernel distribution of convoluted data (top-left) used by our adaptive kernel method and the static Gaussian kernel distribution used by CBS (static-center) **(best viewed in color)**. As an example, the weights of the top-left kernel are taken from layer seven of ResNet18 architecture on the CIFAR100 dataset, where the σ=1.2265 and μ=-0.3461 are recorded at epoch 13. The static Gaussian kernel distribution has a fixed σ = 1 and μ = 1. The darker color means the higher weight, and the lighter one means the less weight.

operation. By annealing standard deviation of the Gaussian kernel (i.e., the variance parameter) CBS controls the amount of high-frequency information that filters in the early stage of training. The adaptive kernel designed by CBS fundamentally improves the ability of CNNs to learn better representation from data.

The adaptive kernel introduced by CBS has less flexibility as they use a fixed mean (μ) and standard deviation (σ) for a specified range of epochs. CBS also uses the same adaptive kernel for every layer of a CNN architecture. This arises three research questions: 1. Can we dynamically initialize the value of deterministic kernel instead of static design done by CBS? 2. Is there any impact of weight distribution in deterministic Gaussian kernel? 3. Is a different or same deterministic adaptive kernel better for each layer of a CNN architecture?

To answer these questions, we propose a dynamic Gaussian distribution based kernel while CBS uses a static Gaussian distribution base kernel. We compute the mean (μ) and standard deviation (σ) of the convoluted data and use them to form a Gaussian kernel (we call it 'Seed kernel'). We create multiple kernel sets by re-distributing the weights from the seed kernel and propose an adaptive kernel selection method to answer questions two and three. Finally, we derive a lemma that shows the representativeness of an adaptive kernel based on the variance parameter. The adaptive Gaussian kernel used in our work is a deterministic function, meaning the kernel's weights are not updated via back-propagation.

Table 1 shows the proposed dynamic seed kernel and the static kernel used by CBS. We name the kernel constructed based on the Gaussian distribution of the convoluted data as top-left (left two matrices in Table 1) because the maximum weight is on the top-left position of the kernel. The static Gaussian distribution (right two matrices in Table 1) holds the maximum weight in the middle of the kernel, and then the weight intensity decreases around the neighbor. The weight distribution nature of the static Gaussian can be described as a ripple effect on the water. However, the weight intensity decreases from one corner to another corner in the Gaussian distribution of convoluted data. We can compare this scenario to painting a wall with a brush where the intensity decreases as the brush moves far from its starting point.

Figure 1 shows the detail of the kernel set creation and selection process by utilizing convoluted data. The top dotted box represents the conventional CNN training phase. The bottom dotted box shows the adaptive kernel creation phase. At each epoch and each layer, our proposed method constructs a Gaussian distribution of convoluted data to form a kernel. After constructing the initial kernel, we re-distribute the weights into kernel space by shifting weights to a different position to create a set of kernels (examples of kernel sets are shown in Table 3). Each layer randomly selects a kernel from the set of kernels and performs a convolution operation on that layer's data and randomly selected kernel. Based on the experimental result, we find out the importance of weight distribution in kernel space and recommend the different deterministic adaptive kernel that can gain better efficiency in a CNN variant.

The main contribution of this paper can be summarized as follows:

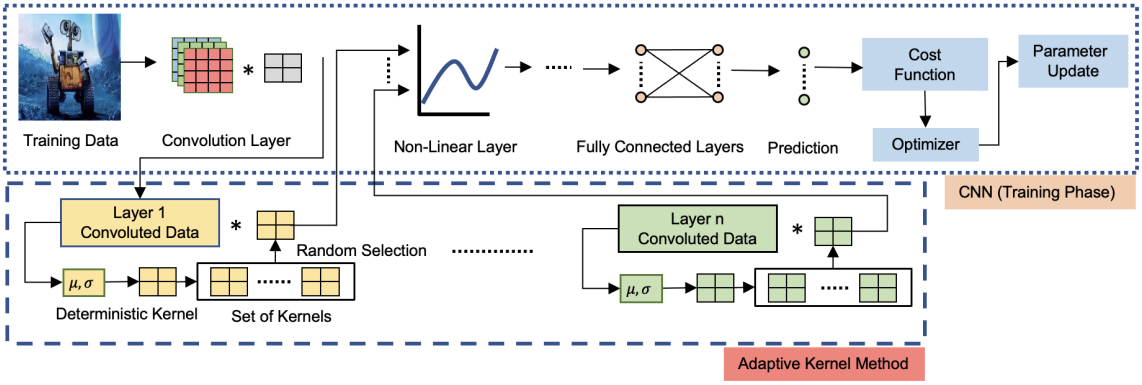


Figure 1: Overview of the adaptive kernel method in Convolutional Neural Network (CNN). We compute μ and σ of a layer's convoluted data to construct a Gaussian kernel ('seed kernel'). Then after shifting weights, we construct a set of kernels (details of this step are in Section 3.2). A random kernel is selected from the set of kernels at each layer. Then a convolution operation on convoluted data and the selected kernel is performed. The outcome of the convolution operation passes to the next component of CNN.

- We create an adaptive kernel from the Gaussian distribution of the layer's convoluted data.
- We re-distribute the weights of the initially created seed kernel into different positions to create a set of adaptive kernels.
- We derive a lemma that shows the representativeness of adaptive kernel based on variance parameter.
- The adaptive kernel designed by our method is a deterministic function and does not add any trainable parameter to the model.
- The implementation of the proposed method (bottom dotted box in Figure 1) can easily be attached to any existing CNN variant (top dotted box in Figure 1).

2 Related Work

2.1 Kernel Design in CNN

A kernel is a matrix of weights that performs convolution operations with input data to extract relevant features. Saxe et al. (2011); Pinto et al. (2011); Jarrett et al. (2009); Salehinejad et al. (2022) gain better result by randomly initialized kernel. To detect edges at a certain orientation or scale, (Gao et al., 2010; Canny, 1986) use static kernel. Coates et al. (2011) use a k-means clustering algorithm to learn centroid as a convolution kernel in small image patches as unsupervised learning. Cho & Saul (2009) use positive-definite kernel functions for shallow and deep kernel-based CNN architectures. Mairal et al. (2014) introduce an unsupervised convolutional neural network that is trained to approximate the kernel map. Bo et al. (2010) introduces three types of kernels to measure similarities between image patches instead of using one kernel.

2.2 Gaussian Kernel in CNN

Gaussian kernel considerably uses in computer vision (Khumaidi et al., 2017; Bouboulis et al., 2010; Sotak Jr & Boyer, 1989; Wang et al., 2003). Gaussian kernel is used as a low-pass filter in signal processing

domain (Deng & Cahill, 1993; Shin et al., 2005; Young & Van Vliet, 1995). Recently, Bietti & Mairal (2019); Lee et al. (2020); Mairal (2016) use the anti-aliasing properties of Gaussian kernel in CNN.

2.3 ADAPTIVE KERNEL DESIGN IN CNN

Adaptive kernel uses dynamic kernels instead of fixed kernel (Du et al., 2017; Li et al., 2017; Xiong et al., 2015). (Li et al., 2015) propose a kernel adaptation method that dynamically determines the convolutional kernels according to the spatial distribution of facial landmarks. Ding et al. (2018); Li et al. (2019) use kernel selection scheme to automatically adjust the receptive field size based on the input. Ding et al. (2017) splits the training data into the cluster and learns an exclusive kernel for each cluster. (Li et al., 2015) dynamically determines the convolutional kernels according to the spatial distribution of facial landmarks, which performs better in learning features. Ding et al. (2017); Jia et al. (2016); Klein et al. (2015); Su et al. (2019); Zamora Esquivel et al. (2019) feed input images into kernel function to dynamically generate kernels. Gong et al. (2021) proposed a new adaptation mechanism that automatically determines the layers to employ dynamic kernel and attention maps.

Curriculum by Smoothing (henceforth, CBS) (Sinha et al., 2020) is a recent work that designs an adaptive kernel for CNN by using a Gaussian kernel. First, fixed values of σ and μ ($\sigma = 1$, $\mu = 1$) are being used to generate a Gaussian kernel. Second, to adapt curricula with the progression of the models' training, they reduce the value of σ by $\sigma \leftarrow \sigma * 0.925$ to create a new Gaussian kernel. Third, they construct a new Gaussian kernel after every five or ten epochs depending on the CNN variants. However, curricula propose by Sinha et al. (2020) is pre-determined and does not learn from the data. It is unclear why they always reduce the σ by $\sigma \leftarrow \sigma * 0.925$ after every five or ten epochs to construct a new Gaussian kernel.

3 CONVOLUTIONAL NEURAL NETWORK (CNN)

To denote the convolutional operation of some kernel θ_w on some input X, we use $\theta_w \circledast X$. In deep learning, a typical CNN is composed of stacked trainable convolutional layers LeCun et al. (1998), pooling layers Boureau et al. (2010), and non-linearities Nair & Hinton (2010).

The input tensor X is organised by batch size, channel number, height, and width. A typical CNN convolution operation at n-th layer can be mathematically represented by Equation 1, where θ_w are the learned weights of the kernel.

$$X_n = (\theta_w \circledast X_{n-1}) \tag{1}$$

3.1 GAUSSIAN KERNEL

We use a parameterized Gaussian kernel after the convolution operation. This Gaussian kernel is analogous to a convolutional layer kernel. Bietti & Mairal (2019); Lee et al. (2020); Mairal (2016) use the anti-aliasing properties of Gaussian kernel in CNN. Standard deviation (σ) is the hyperparameter of the kernel, which masks the high-frequency information from the input by using Gaussian's low-pass filter attribute. x has a Gaussian distribution with parameters μ and σ, denoted by

$$\theta_G(x;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}, x \in \mathbb{R} \tag{2}$$

where $\mu \in \mathbb{R}$ and $\sigma > 0$.

.3315	.1889	.0554
.1889	.1076	.0315
.0554	.0315	.0092

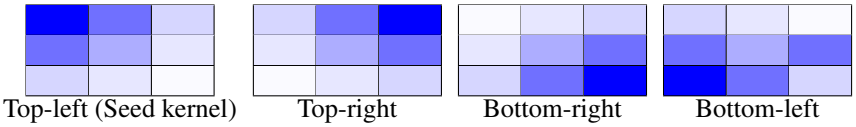


Table 2: The base four elements of Gaussian kernels for our adaptive kernel set **(best viewed in color)**. As an example, the values of the top-left kernel (i.e., the 'Seed kernel') are taken from layer seven of ResNet18 architecture on the CIFAR100 dataset, where the σ=1.2265 and μ=-0.3461 are recorded at epoch 13. The 'top-left' kernel is always constructed from the convoluted data. The rest of the kernels are constructed by shifting weights to different positions by clockwise. The intensity of the color represents the weight. The darker color means the higher weight, and the lighter one means the less weight.

A 2D Gaussian kernel can be constructed as a deterministic function of the size of the kernel by the following equation

$$\theta_G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{3}$$

3.2 Design of Adaptive Gaussian Kernel

The Gaussian kernel is used to the output of convolutional layer to train a CNN variant via curriculum learning Sinha et al. (2020). The addition (i.e., the convolution operation) of the Gaussian kernel to the CNN layer can be expressed by the following equation:

$$h_n = ReLU(pool(\theta_G \circledast (\theta_w \circledast X_n))) \tag{4}$$

where h_n is the n-th hidden layer and θ_G is the Gaussian kernel. We compute the σ and μ from the convoluted data and determine θ_G by using Equation 3. We perform convolution operation on convoluted data (i.e., $\theta_w \circledast X_n$) and the Gaussian kernel (i.e., θ_G) as shown in Equation 4. It is noteworthy that we use a deterministic Gaussian kernel so that the Gaussian kernel is not trained via backpropagation.

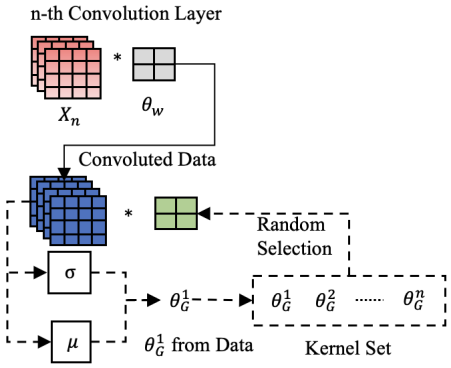


Figure 2: Gaussian kernel construction process from the convoluted data. After constructing the seed Gaussian kernel (θ_G^1) from the convoluted data, a set of kernels are also constructed by shifting weights.

We compute a set of adaptive kernels at each layer of a CNN variant as shown in Figure 1. Figure 2 shows the process of computing the set of kernels for a particular layer of a CNN variant. The exact process has been applied to all the layers of a CNN variant. A random adaptive kernel is chosen from the kernel set at each layer and the convolution operation is performed on the convoluted data.

First, we compute the σ and μ from the convoluted data of a layer (Figure 2). Based on the σ and μ, we compute the Gaussian Kernel (θ_G) using Equation 3. We call it the 'Seed Gaussian Kernel' and denote it by θ_G^1 (the left kernel shown in Figure 2). The 'top-left' labeled kernel in Figure 2 shows the color representation of θ_G^1. θ_G^1 always put maximum weight on the top-left position of the kernel. We compute three more kernels from θ_G^1 by shifting weights to different positions of the kernel. For example, to get 'top-right' kernel (i.e., θ_G^2, which puts maximum weight on the top right position of the kernel), we move the first column of θ_G^1 to the first row of θ_G^2, the second column of θ_G^1 to the second row of θ_G^2, and the third column of θ_G^1 to the

third row of θ_G^2, all clockwise. All these four kernels (i.e., θ_G^1, θ_G^2, θ_G^3, θ_G^4 in Figure 2) are the base elements of the kernel set. Different combination of these four kernels are used to form different kernel sets. In our adaptive kernel design the following five types of kernel sets:

- Set of one kernel (1-Kernel): {top-left} — {bottom-right} — {top-right} — {bottom-left}
- Set of two kernels (2-Kernel): {top-left, bottom-right} — {bottom-left, top-right} — {top-left, bottom-left} — {top-right, bottom-right} — {top-left, top-right} — {bottom-left, bottom-right}
- Set of three kernels (3-Kernel): {top-left, bottom-right, static-center} — {bottom-left, top-right, static-center} — {top-left, bottom-left, static-center} — {top-right, bottom-right, static-center} — {top-left, top-right, static-center} — {bottom-left, bottom-right, static-center}
- Set of four kernels (4-Kernel): {top-left, bottom-right, top-right, bottom-left}
- Set of five kernels (5-Kernel): {top-left, bottom-right, top-right, bottom-left, static-center}

3.3 Adaptive Kernel Selection Lemma

Let the total number of kernel sets be k and our goal is to find the kernel set number $\hat{k} \in T$ where $T = \{y : k \geq y \geq 1\}$ that maximize the accuracy/performance. In each kernel set (we call it 'Type'), theoretically, there could be any number of square matrices. Let the number of square matrices (of n dimension) of a kernel set j be m_j. The direct sum of square matrices $A_1, A_2, \ldots, A_m$ (of equal dimension) is defined recursively as follows: if $A_1, A_2 \in \mathbb{R}^{n \times n}$, the direct sum of A_1 and A_2 is given by

$$A_1 \oplus A_2 = B \in \mathbb{R}^{n \times n}, B_{ij} = (A_1)_{ij} + (A_2)_{ij} \text{ for } 1 \leq i, j \leq n$$

The definition then extends recursively,

$$\begin{aligned} \bigoplus_{i=1}^{m} A_i &= A_1 \oplus A_2 \oplus \cdots \oplus A_m \\ &:= (A_1 \oplus A_2 \oplus \cdots \oplus A_{m-1}) \oplus A_m \end{aligned}$$

The standard deviation of the direct sum of square matrices $A_1, A_2, \ldots, A_m$ of a kernel set j is defined as follows:

$$\sigma_j(\bigoplus_{i=1}^{m} A_i) \text{ where } j \in T$$

The desired kernel set would be the one with minimum standard deviation of the direct sum of square matrices in it and is defined as follows:

$$\hat{k} = \arg\min_{j \in T}(\sigma_j(\bigoplus_{i=1}^{m} A_i))$$

4 Evaluation and Experimental Results

In this section, we empirically evaluate the effectiveness of our adaptive kernel method on three different CNN variants. We test these CNN variants on three different datasets (i.e., CIFAR10, CIFAR100 (Krizhevsky et al., 2009), and SVHN (Netzer et al., 2011)) and report Top-1 classification accuracy. For optimization, we use stochastic gradient descent (SGD) with the same learning rate scheduling, momentum, and weight decay as stated in the original papers (Sinha et al., 2020; He et al., 2016; Simonyan & Zisserman, 2014), without hyper-parameter tuning. The objective of all image classification experiments is a standard unweighted multi-class cross-entropy loss (Sinha et al., 2020).

4.1 CNN Variants, datasets, tasks and Evaluation Metrics

To evaluate our adaptive kernel selection method, we perform the image classification task on two standard vision datasets, CIFAR10 and CIFAR100, containing images of 10 and 100 classes, respectively. SVHN,

Type	Kernel Set	Superposition	σ	Group
1	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right)	.340 .220 .110 / .220 .251 .220 / .110 .220 .340	0.07671	3
2	.055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left) ; .055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right)	.110 .220 .340 / .220 .251 .220 / .340 .220 .110	0.07671	
3	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left)	.386 .220 .014 / .377 .215 .063 / .386 .220 .014	0.14486	5
4	.055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right)	.014 .220 .386 / .063 .215 .377 / .014 .220 .386	0.14486	
5	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right)	.386 .377 .386 / .220 .215 .220 / .014 .063 .014	0.14486	
6	.055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right)	.014 .063 .014 / .220 .215 .220 / .386 .377 .386	0.14486	
7	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right) ; .075 .123 .075 / .123 .204 .123 / .075 .123 .075 (Static-center)	.415 .344 .185 / .344 .419 .344 / .185 .344 .415	0.08495	4
8	.055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left) ; .055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right) ; .075 .123 .075 / .123 .204 .123 / .075 .123 .075 (Static-center)	.185 .344 .415 / .344 .419 .344 / .415 .344 .185	0.08495	
9	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left) ; .075 .123 .075 / .123 .204 .123 / .075 .123 .075 (Static-center)	.462 .344 .089 / .501 .419 .186 / .462 .344 .089	0.15204	6
10	.055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right) ; .075 .123 .075 / .123 .204 .123 / .075 .123 .075 (Static-center)	.089 .344 .462 / .186 .419 .501 / .089 .344 .462	0.15204	
11	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right) ; .075 .123 .075 / .123 .204 .123 / .075 .123 .075 (Static-center)	.462 .501 .462 / .344 .419 .344 / .089 .186 .089	0.15204	
12	.055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right) ; .075 .123 .075 / .123 .204 .123 / .075 .123 .075 (Static-center)	.089 .186 .089 / .344 .419 .344 / .462 .501 .462	0.15204	
13	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right) ; .055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left) ; .055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right)	.451 .440 .451 / .440 .430 .440 / .451 .440 .451	**0.00707**	**1**
14	.331 .188 .055 / .188 .107 .031 / .055 .031 .009 (Top-left) ; .009 .013 .055 / .031 .107 .188 / .055 .188 .331 (Bottom-right) ; .055 .031 .009 / .188 .107 .031 / .331 .188 .055 (Bottom-left) ; .055 .188 .331 / .031 .107 .188 / .009 .031 .055 (Top-right) ; .075 .123 .075 / .123 .204 .123 / .075 .123 .075 (Static-center)	.526 .564 .526 / .564 .634 .564 / .526 .564 .526	0.03320	2

Table 3: The weights of the kernel are taken from layer seven of ResNet18 architecture on the CIFAR100 dataset, where the σ=1.2265 and μ=-0.3461 are recorded at epoch 13. The static Gaussian kernel distribution has a fixed $\sigma = 1$ and $\mu = 1$. The superposition kernel is constructed by the lemma defined in Section 3.3. The darker color means the higher weight, and the lighter one means the less weight. The second last column shows the standard deviation value of the superposition kernel. The last column shows the ranked group (number) of kernel type(s) based on the increasing values of the lemma's standard deviation (σ).

Table 4: Top-1 classification accuracy (Acc.) on CIFAR10, CIFAR100, and SVHN datasets. The **bold** numbers represent the better scores.

Model		Dataset		
		CIFAR 10	CIFAR 100	SVHN
Base	CNN	80.4 ± 0.2	47.2 ± 0.2	89.4 ± 0.2
	ResNet18	89.3 ± 0.3	64.3 ± 0.3	95.0 ± 0.3
	VGG16	82.0 ± 0.3	48.8 ± 0.3	93.8 ± 0.3
Base + CBS	CNN + CBS	77.3 ± 0.2	46.5 ± 0.2	89.4 ± 0.2
	ResNet18 + CBS	89.3 ± 0.2	65.8 ± 0.3	96.1 ± 0.3
	VGG16 + CBS	83.6 ± 0.3	49.1 ± 0.3	94.2 ± 0.3
Base + 1 Kernel Set	CNN +1-Kernel	78.4 ± 0.2	47.2 ± 0.3	89.4 ± 0.2
	ResNet18+ 1-Kernel	87.7 ± 0.3	61.3 ± 0.3	93.3 ± 0.3
	VGG16 + 1-Kernel	81.2 ± 0.3	46.1 ± 0.3	88.5 ± 0.3
Base + 2 Kernel Set	CNN + Kernel Type-1	83.5 ± 0.2	47.4 ± 0.2	89.4 ± 0.2
	ResNet18 + Kernel Type-1	89.7 ± 0.3	68.1 ± 0.3	95.7 ± 0.3
	VGG16 + Kernel Type-1	83.5 ± 0.3	51.8 ± 0.3	94.4 ± 0.3
	CNN + Kernel Type-2	83.4 ± 0.2	47.4 ± 0.2	89.4 ± 0.2
	ResNet18 + Kernel Type-2	89.7 ± 0.3	68.8 ± 0.3	95.7 ± 0.3
	VGG16 + Kernel Type-2	83.5 ± 0.3	51.6 ± 0.3	94.4 ± 0.3
	CNN + Kernel Type-3	81.7 ± 0.2	47.2 ± 0.2	89.4 ± 0.2
	ResNet18 + Kernel Type-3	87.1 ± 0.3	62.6 ± 0.3	95.8 ± 0.3
	VGG16 + Kernel Type-3	83.6 ± 0.3	50.9 ± 0.3	92.3 ± 0.3
	CNN + Kernel Type-4	80.6 ± 0.2	46.9 ± 0.2	88.4 ± 0.2
	ResNet18 + Kernel Type-4	89.2 ± 0.3	64.2 ± 0.3	95.9 ± 0.3
	VGG16 + Kernel Type-4	83.3 ± 0.3	50.6 ± 0.3	91.0 ± 0.3
	CNN + Kernel Type-5	81.9 ± 0.2	45.9 ± 0.2	89.2 ± 0.2
	ResNet18 + Kernel Type-5	87.9 ± 0.3	61.7 ± 0.3	95.6 ± 0.3
	VGG16 + Kernel Type-5	83.5 ± 0.3	50.3 ± 0.3	89.8 ± 0.3
	CNN + Kernel Type-6	81.2 ± 0.2	46.1 ± 0.2	89.0 ± 0.2
	ResNet18 + Kernel Type-6	89.6 ± 0.3	64.2 ± 0.3	95.7 ± 0.3
	VGG16 + Kernel Type-6	83.6 ± 0.3	50.3 ± 0.3	89.2 ± 0.3
Base + 3 Kernel Set	CNN + Kernel Type-7	83.0 ± 0.2	47.3 ± 0.2	89.4 ± 0.2
	ResNet18 + Kernel Type-7	89.2 ± 0.3	65.8 ± 0.3	95.8 ± 0.3
	VGG16 + Kernel Type-7	84.8 ± 0.3	51.1 ± 0.3	94.2 ± 0.3
	CNN + Kernel Type-8	81.6 ± 0.2	47.1 ± 0.2	89.4 ± 0.2
	ResNet18 + Kernel Type-8	90.0 ± 0.3	65.5 ± 0.3	95.8 ± 0.3
	VGG16 + Kernel Type-8	84.5 ± 0.3	51.3 ± 0.3	94.2 ± 0.3
	CNN + Kernel Type-9	81.6 ± 0.2	47.2 ± 0.2	89.4 ± 0.2
	ResNet18 + Kernel Type-9	87.1 ± 0.3	62.5 ± 0.3	95.7 ± 0.3
	VGG16 + Kernel Type-9	82.5 ± 0.3	50.7 ± 0.3	92.2 ± 0.3
	CNN + Kernel Type-10	80.4 ± 0.2	46.5 ± 0.2	88.4 ± 0.2
	ResNet18 + Kernel Type-10	89.1 ± 0.3	64.1 ± 0.3	95.6 ± 0.3
	VGG16 + Kernel Type-10	82.2 ± 0.3	50.2 ± 0.3	90.5 ± 0.3
	CNN + Kernel Type-11	81.7 ± 0.2	45.8 ± 0.2	89.2 ± 0.2
	ResNet18 + Kernel Type-11	87.7 ± 0.3	61.6 ± 0.3	95.5 ± 0.3
	VGG16 + Kernel Type-11	82.3 ± 0.3	50.2 ± 0.3	89.8 ± 0.3
	CNN + Kernel Type-12	81.0 ± 0.2	46.1 ± 0.2	88.9 ± 0.2
	ResNet18 + Kernel Type-12	89.5 ± 0.3	64.2 ± 0.3	95.5 ± 0.3
	VGG16 + Kernel Type-12	82.6 ± 0.3	50.2 ± 0.3	89.2 ± 0.3
Base + 4 Kernel Set	CNN + Kernel Type-13	**83.5 ± 0.2**	**47.6 ± 0.2**	**89.5 ± 0.2**
	ResNet18 + Kernel Type-13	**91.7 ± 0.2**	**69.0 ± 0.3**	**96.7 ± 0.3**
	VGG16 + Kernel Type-13	**85.7 ± 0.2**	**52.3 ± 0.3**	**94.7 ± 0.3**
Base + 5 Kernel Set	CNN + Kernel Type-14	83.0 ± 0.2	47.2 ± 0.2	89.4 ± 0.3
	ResNet18 + Kernel Type-14	90.0 ± 0.2	66.3 ± 0.2	96.4 ± 0.3
	VGG16 + Kernel Type-14	85.1 ± 0.2	51.5 ± 0.3	94.5 ± 0.3

the other dataset, is a digit recognition dataset that consists of natural images of the 10 digits collected from the street view. Both ResNet (He et al., 2016) and VGG (Simonyan & Zisserman, 2014) are based on CNN architecture and have different variations based on the number of layers. We consider ResNet18 (He et al., 2016) and VGG16 (Simonyan & Zisserman, 2014) variations in our experiment. Curriculum by Smoothing (CBS) (Sinha et al., 2020) uses an adaptive kernel to train a CNN variant. CBS applied to CNN variants, such as ResNet18 + CBS and VGG16 + CBS, to improve the accuracy of image classification tasks. We compare our adaptive kernel method with the base models (i.e., CNN, ResNet18, and VGG16), CBS and reported kernel set variations for the Top-1 classification accuracy and Pearson correlation for the three mentioned datasets.

4.2 Results and Analysis

The experimental results are summarized in Table 4. We run experiments with and without our adaptive kernel to evaluate Top-1 classification accuracy (Acc.). Based on our experiment, 'Kernel Type-13' (top-left, top-right, bottom-right, bottom-left) outperformed the base, CBS, and all the other variations of kernels in CIFAR10, CIFAR100, and SVHN datasets.

	Dataset		
Model	CIFAR10	CIFAR100	SVHN
CNN	0.8727	0.8230	0.8155
ResNet18	0.8589	0.8421	0.8916
VGG16	0.8871	0.8887	0.8832

Table 5: Pearson correlation between the ranked Group numbers and their Top-1 classification accuracy for different CNN variants and datasets.

We can sort the fourteen types of kernels into six groups based on the increasing values of the lemma's standard deviation (σ). The groups are as follows: Group 1=$\{$Kernel Type 13$\}$, Group 2=$\{$Kernel Type 14$\}$, Group 3=$\{$Kernel Type 1, 2$\}$, Group 4=$\{$Kernel Type 7, 8$\}$, Group 5=$\{$Kernel Type 3, 4, 5, 6$\}$, and Group=6 $\{$Kernel Type 9, 10, 11, 12$\}$. Group 1 (i.e., Kernel Type 13) has the lowest σ, and our lemma predicts that Kernel Type 13 is better than the other kernel types. Similarly, Kernel Type 14 should perform better than Kernel Type 7 and Kernel Type 8. At the same time, our lemma predicts that Kernel Type 7 and Kernel Type 8 should have similar classification accuracy as they are in the same group. We compute the *Pearson correlations* (shown in Table 5) between the ranked Group numbers and their Top-1 classification accuracy for different CNN variants and datasets to verify if the classification accuracies of all the six groups of kernels maintain the order predicted by our lemma.

Static Gaussian distribution has more weights in the center, and the weight intensity decreases from the center. The drawback of this approach is the center part of the data always gains higher weight and neglects the feature property around the border. In contrast to static design, the proposed adaptive kernel method has an equal probability of choosing a kernel from the kernel type at each epoch in training. As a result, the adaptive kernel method can collectively emphasize all data positions in training. Collectively, the lower the σ, the higher the uniformity of weights in kernel space. The higher uniformity of weights in kernel space emphasizes all data positions in convolution operation.

5 Conclusion

In this research work, we dynamically initialize a deterministic adaptive kernel. We construct a 'Seed' kernel based on the Gaussian distribution of convoluted data. We redistribute the weights of the seed kernel and construct a set of kernel types. We also derived a lemma that shows the 'quality' of the representativeness of kernel type. Based on our experiment, a set of four kernels perform better compared to other kernel variations. It indicates that the convoluted data distribution in kernel space has an impact. Kernel type with minimum standard deviation performs better as it has higher uniformity in kernel space that emphasizes all data positions in convolution operation. Our adaptive kernel method achieves better accuracy than the CBS by the range of 0.1%$\sim$3.2% in image classification task.

REFERENCES

Alberto Bietti and Julien Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *The Journal of Machine Learning Research*, 20(1):876–924, 2019.

Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. *Advances in neural information processing systems*, 23, 2010.

Pantelis Bouboulis, Konstantinos Slavakis, and Sergios Theodoridis. Adaptive kernel-based image denoising employing semi-parametric regularization. *IEEE Transactions on Image Processing*, 19(6):1465–1479, 2010.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Guang Deng and LW Cahill. An adaptive gaussian filter for noise reduction and edge detection. In *1993 IEEE conference record nuclear science symposium and medical imaging conference*, pp. 1615–1619. IEEE, 1993.

Chen Ding, Ying Li, Yong Xia, Wei Wei, Lei Zhang, and Yanning Zhang. Convolutional neural networks based hyperspectral image classification method with adaptive kernels. *Remote Sensing*, 9(6):618, 2017.

Chen Ding, Ying Li, Yong Xia, Lei Zhang, and Yanning Zhang. Automatic kernel size determination for deep neural networks based hyperspectral image classification. *Remote Sensing*, 10(3):415, 2018.

Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

Wenshuo Gao, Xiaoguang Zhang, Lei Yang, and Huizhong Liu. An improved sobel edge detection. In *2010 3rd International conference on computer science and information technology*, volume 5, pp. 67–71. IEEE, 2010.

Sixue Gong, Xiaoming Liu, and Anil K Jain. Mitigating face recognition bias via group adaptive classifier. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3414–3424, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pp. 2146–2153. IEEE, 2009.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016.

Agus Khumaidi, Eko Mulyanto Yuniarno, and Mauridhi Hery Purnomo. Welding defect classification based on convolution neural network (cnn) and gaussian kernel. In *2017 international seminar on intelligent technology and its applications (ISITIA)*, pp. 261–265. IEEE, 2017.

Benjamin Klein, Lior Wolf, and Yehuda Afek. A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4840–4848, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jungkyu Lee, Taeryun Won, Tae Kwan Lee, Hyemin Lee, Geonmo Gu, and Kiho Hong. Compounding the performance improvements of assembled techniques in a convolutional neural network. *arXiv preprint arXiv:2001.06268*, 2020.

Shaoxin Li, Junliang Xing, Zhiheng Niu, Shiguang Shan, and Shuicheng Yan. Shape driven kernel adaptation in convolutional neural network for robust facial traits recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 222–230, 2015.

Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 510–519, 2019.

Xudong Li, Mao Ye, Yiguang Liu, and Ce Zhu. Adaptive deep convolutional neural networks for scene-specific object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(9):2538–2551, 2017.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Julien Mairal. End-to-end kernel learning with supervised convolutional kernel networks. *Advances in neural information processing systems*, 29, 2016.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. *Advances in neural information processing systems*, 27, 2014.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Nicolas Pinto, Zak Stone, Todd Zickler, and David Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *CVPR 2011 WORKSHOPS*, pp. 35–42. IEEE, 2011.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Hojjat Salehinejad, Yang Wang, Yuanhao Yu, Tang Jin, and Shahrokh Valaee. S-rocket: Selective random convolution kernels for time series classification. *arXiv preprint arXiv:2203.03445*, 2022.

Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *Icml*, 2011.

Dong-Hyuk Shin, Rae-Hong Park, Seungjoon Yang, and Jae-Han Jung. Block-based noise estimation using adaptive gaussian filtering. *IEEE Transactions on Consumer Electronics*, 51(1):218–226, 2005.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Samarth Sinha, Animesh Garg, and Hugo Larochelle. Curriculum by smoothing. *Advances in Neural Information Processing Systems*, 33, 2020.

George E Sotak Jr and Kim L Boyer. The laplacian-of-gaussian kernel: a formal analysis and design procedure for fast, accurate convolution and full-frame output. *Computer Vision, Graphics, and Image Processing*, 48(2):147–189, 1989.

Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11166–11175, 2019.

Wenjian Wang, Zongben Xu, Weizhen Lu, and Xiaoyun Zhang. Determination of the spread parameter in the gaussian kernel for classification and regression. *Neurocomputing*, 55(3-4):643–663, 2003.

Chao Xiong, Xiaowei Zhao, Danhang Tang, Karlekar Jayashree, Shuicheng Yan, and Tae-Kyun Kim. Conditional convolutional neural network for modality-aware face recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3667–3675, 2015.

Ian T Young and Lucas J Van Vliet. Recursive implementation of the gaussian filter. *Signal processing*, 44 (2):139–151, 1995.

Julio Zamora Esquivel, Adan Cruz Vargas, Paulo Lopez Meyer, and Omesh Tickoo. Adaptive convolutional kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

A Appendix

You may include other additional sections here.