LOG-POLAR SPACE CONVOLUTION LAYERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional neural networks use regular quadrilateral convolution kernels to extract features. Since the number of parameters increases quadratically with the size of the convolution kernel, many popular models use small convolution kernels, resulting in small local receptive fields in lower layers. This paper proposes a novel log-polar space convolution (LPSC) layer, where the convolution kernel is elliptical and adaptively divides its local receptive field into different regions according to the relative directions and logarithmic distances. The local receptive field grows exponentially with the number of distance levels. Therefore, the proposed LPSC not only naturally encodes local spatial structures, but also greatly increases the single-layer receptive field while maintaining the number of parameters. We show that LPSC can be implemented with conventional convolution via log-polar space pooling and can be applied in any network architecture to replace conventional convolutions. Experiments on different tasks and datasets demonstrate the effectiveness of the proposed LPSC.

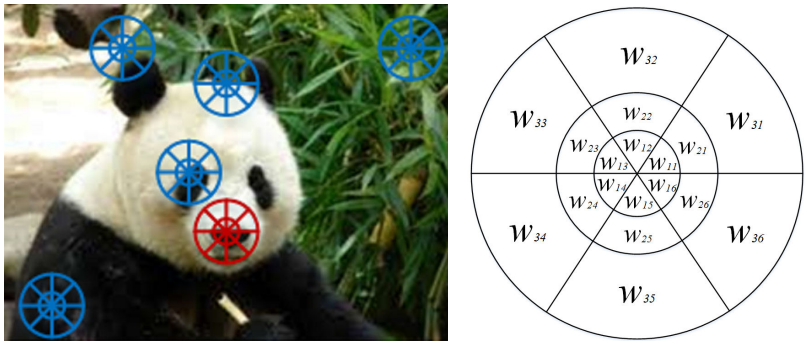
1 INTRODUCTION

Convolutional neural networks (LeCun et al., 1998; Krizhevsky et al., 2012) have achieved great success in the field of computer vision. The size of the convolution kernel determines the locally weighted range of the image or feature map, which is called the *local receptive field (LRF)*. In many computer vision tasks such as image classification and intensive prediction, larger LRF is generally desired to capture the dependencies between long-distance spatial positions and a wide range of context information. Simply increasing the size of the convolution kernel is not plausible because the number of parameters increases quadratically with the size.

In practice, commonly used techniques to obtain larger receptive fields include replacing a single-layer large convolution kernel with multi-layer small convolution kernels, adding pooling layers, and using dilated convolutions (Holschneider et al., 1990; Yu & Koltun, 2015). Increasing the number of convolutional layers may cause vanishing gradients and make training more difficult. The pooling process often causes information loss. Dilated convolution kernels are not continuous since not all pixels in the LRF are involved in convolution calculation. The skipped pixels are regularly selected. With the same number of parameters, the larger the LRF, the more pixels are skipped, which may miss some details and cause discontinuity of information.

Moreover, conventional and dilated convolutions use regular square kernels. Each position is assigned a different weight within the LRF. All positions are equally treated regardless of the size of the kernel. However, intuitively, the correlation between neighboring pixels and the center pixel is usually higher, while the farther the pixel, the smaller the impact on the center pixel. The effects of two adjacent pixels that are far away from the center are usually similar, thus they can share the same parameter rather than be assigned different weights separately. As shown in red in Fig. 1(a), according to the configuration of surrounding regions, it can be inferred that the center position is located on the upper edge of the nose. Pixels in the same upper-left outer half-fan-shaped region show that the far upper left of the center point is white fur, but there is little difference in the effects of two specific fur points.

In this paper, we propose a novel *log-polar space convolution (LPSC)* method. The shape of the LPSC kernel is not a regular square, but an ellipse. Parameters of the kernel are not evenly distributed in the LRF, but are assigned in the log-polar coordinate space. As shown in Fig. 1(b), the LPSC kernel divides the LRF into different regions, where regions become larger with the increase of the



(a) (b)

Figure 1: (a) At different locations of an image, local contextual pixels can be divided into different regions according to their relative distances and directions in the log-polar space. For each location, pixels falling in the same region are generally similar and can share the same weight. (b) The LPSC kernel. For this example, $L_r = 3, L_\theta = 6, g = 2$, thus there are only 18+1 parameters. The LRF can be arbitrarily large.

distance to the center. Pixels that fall into the same region share the same weight. In this way, LPSC can increase the LRF exponentially without increasing the number of parameters. Besides, LPSC naturally imposes a contextual structure on the local neighboring distribution.

The main contributions of this paper include: 1. We propose a new convolution method where the kernel lies in the log-polar space to capture the structured context information and greatly expand the LRF without increasing the number of parameters. 2. We propose log-polar space pooling to up-sample the feature map, by which conventional convolution can be conveniently used to achieve LPSC. 3. We apply LPSC to replace the conventional and dilated convolution in different network architectures including AlexNet, VGGNet, ResNet, DeepLabv3+, and CE-Net. We demonstrate the effectiveness of LPSC through empirical evaluations on different tasks and datasets.

2 Related work

Context pooling. Our method is highly motivated by shape context (Belongie et al., 2001; 2002). Centered at a reference point, all other points are divided into bins that are uniformly distributed in the log-polar space. The histogram among these bins is used as the descriptor. Geometric blur (Berg & Malik, 2001) sparsely samples and aggregates a blurred signal in the log-polar space. Pyramid context (Barron et al., 2013) pools log-spaced context points at multiple scales. Different from these methods, we design a kernel in the log-polar space for convolution, each region is assigned a weight to aggregate information from the bins. We incorporate the kernel into deep neural networks.

Methods to increase LRFs. In Simonyan & Zisserman (2014) and He et al. (2016), it is found that imposing a regularization on large convolution kernels is equivalent to the superposition of multiple convolution layers with smaller kernels. Based on this observation, many state-of-the-art network architectures use multi-layer small kernels. However, deeper layers may cause vanishing gradients, making the network more difficult to train. Moreover, according to Luo et al. (2016), the effective receptive field (ERF) is proportional to the square root of the depth and proportional to the kernel size. Thus it is easier to achieve a large ERF by increasing the kernel size than by adding layers. We provide a way to increase the LRF without increasing either the number of layers or the number of parameters. In cases where large input or LRF is required but very deep networks are not allowed restricted by resources, our method may be applied to construct a lightweight model.

In Holschneider et al. (1990), atrous (or dilated) convolution increases the LRF by inserting holes (zeros) between parameters in the kernel, where the interval is determined by a dilation rate. Dilated convolution has been applied in different tasks (Dai et al., 2016; Yu & Koltun, 2015; Chen et al., 2017a; Sevilla-Lara et al., 2016; Yang et al., 2018; Chen et al., 2018a). In Zhang et al. (2017a) and Zhang et al. (2019), scale-adaptive convolution learns adaptive dilation rate with a scale regression layer. Due to the insertion of holes, not all pixels in the LRF are used for calculating the output.

In Wang et al. (2018) and Wu et al. (2019), this problem is alleviated by hybrid dilated convolution and Kronecker convolution that uses the Kronecker product to share parameters.

Other convolution methods. Fractionally strided convolution (Zeiler et al., 2010; 2011) up-samples the input by padding. In Jaderberg et al. (2015), a spatial transformer transforms the regular spatial grid into a sampling grid. Active convolution (Jeon & Kim, 2017) learns the shape of convolution by introducing the convolution unit with position parameters. Deformable convolution (Dai et al., 2017) learns additional offsets to augment the sampling locations, thereby adaptively changing the LRF into a polygon. For active and deformable convolutions, the adapted LRF contains holes, the positions and offsets are learned through additional convolutions, which increases the parameters. Deformable kernels (Gao et al., 2019) resample the original kernel space and adapt it to the deformation of objects. The offsets for kernel positions also need to be learned.

Group convolution (Krizhevsky et al., 2012; Zhang et al., 2017b; 2018) and separable convolution (Chollet, 2017) do not increase the LRF of kernels. Octave convolution (Chen et al., 2019) decomposes the feature map into high-frequency and low-frequency features. Multi-scale convolution is performed in Peng et al. (2019) and Li et al. (2020). In Ramachandran et al. (2019) and Cordonnier et al. (2019), stand-alone self-attention is used to replace convolution. The filter in the attention module also lies in a regular and square grid. In Esteves et al. (2018), the polar transformer network (PTN) generates a log-polar representation of the input by differentiable sampling and interpolation techniques. The polar transform is only applied to a single predicted origin location. In contrast, LPSC performs log-polar pooling via binning and can be applied at any location.

Differences. For dilated and other advanced convolutions, the kernel is still performed in a regular grid and all parameters are treated equally. Regardless of the distance from the center, the interval or the sharing range of a parameter is the same among different positions. In contrast, the proposed LPSC expands the LRF in the log-polar space, where near and far regions are distinguished in parameter sharing. The farther away from the center, the larger the range of parameter sharing.

3 LOG-POLAR SPACE CONVOLUTION

Let $\boldsymbol{X} \in \mathbb{R}^{H \times W \times C}$ be the input image or feature map, where H, W, and C are the height, width, and number of channels of $\boldsymbol{X}$, respectively. $\boldsymbol{W} \in \mathbb{R}^{(2M+1)\times(2N+1)\times C}$ is a conventional convolution kernel with a size of $(2M+1) \times (2N+1)$. The central parameter of $\boldsymbol{W}$ is indexed by $(0, 0)$, parameters of $\boldsymbol{W}$ lie in a regular grid $\{(-M, -N), (-M, -N+1), \cdots, (M-1, N), (M, N)\}$. The convolution operation is performed in the 2D spatial domain across the channels. For a spatial location (i, j), the output of the conventional convolution is calculated as

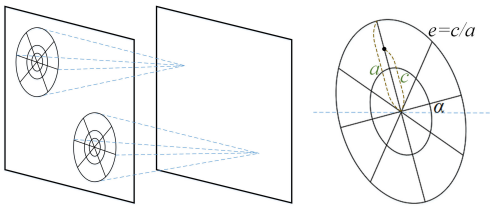
$$(\boldsymbol{X} * \boldsymbol{W})(i, j) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} (\boldsymbol{X}(i+m, j+n) \cdot \boldsymbol{W}(m, n)) + b, \tag{1}$$

where b is the bias. Strictly, Eq. (1) actually performs cross-correlation. For convolution, $\boldsymbol{W}$ needs to be rotated 180 degrees. However, since we can view the learned $\boldsymbol{W}$ as the rotated kernel, we follow the common practice of CNN to formulate convolution into Eq. (1). Parameters of the kernel are uniformly distributed in the regular grid, thus each pixel of $\boldsymbol{X}$ falling into the field is weighted by a separate parameter, i.e., all positions are equally treated. However, pixels that have different distances and directions from the center may have different impacts, e.g., pixels adjacent to the center should have larger contributions to the output. Pixels in the input image usually change gently, adjacent pixels far away from the center often have similar impacts on the center. Based on these intuitions, we design a convolution kernel with a special structure, namely *Log-Polar Space Convolution (LPSC)* kernel, to express a wide range of contextual configurations.

3.1 LPSC KERNEL

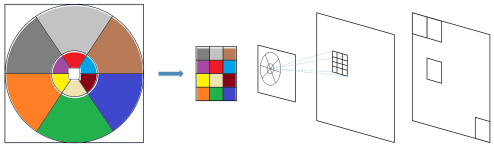
As shown in Fig. 1(b), the proposed LPSC kernel lies in the log-polar space and is shaped by the size $2R+1$, the number of distance levels L_r, the number of direction levels L_θ, and the growth rate g. The LRF of the kernel is the area of the outermost circle whose radius is R. It is uniformly divided into $L_r \times L_\theta$ regions in the log-polar space. Specifically, the log radius is uniformly divided into L_r levels, i.e.,

$$log(R_{l+1}) - log(R_l) = log(R_l) - log(R_{l-1}) = log(g), \tag{2}$$



(a) (b)

Figure 2: (a) The LPSC kernel is slide through the feature map. (b) The shape and inclination of the LPSC kernel can be changed.



(a) (b)

Figure 3: (a) At each position, log-polar space pooling generates a $2L_r \times L_\theta/2$ matrix using the pre-computed mask. (b) To perform LPSC on the original input (left), log-polar pooling generates a matrix for each pixel, resulting in an up-sampled map (middle), conventional convolution with kernel size and stride equaling $(2L_r, L_\theta/2)$ is applied to this map (right).

where $R_l, l = 1, \cdots, L_r$ is the radius of the l-th level and the growth rate g is a hyperparameter controlling the expansion speed. When the center of the kernel is located at position (c_h, c_w), all pixels of $\boldsymbol{X}$ in the range of $\Delta = [c_h - R, c_h + R] \times [c_w - R, c_w + R]$ are divided into L_r levels according to their relative squared distances to the center position. The position $(i, j) \in \Delta$ belongs to the l-th distance level if $R_{l-1} \leq d_{i,j} < R_l$, where $d_{i,j} = (i - c_h)^2 + (j - c_w)^2$. From Eq. (2), we have $R_l = g^{l-1} R_1$. When the innermost radius R_1 is fixed, the LRF grows exponentially with the increase of L_r. The LRF is determined by R which can be set arbitrarily. Given $R_{L_r} = R^2$ and g, we calculate $R_1 = max(2, R^2/g^{L_r-1})$. We use $R = \sqrt{R_{L_r}}$ as a hyperparameter instead of R_1, which is more flexible. Since we use the squared distance, we impose a minimum value of 2 to ensure that all 8-neighborhood pixels fall into the 1-st level.

All positions in the range of Δ are also uniformly divided into L_θ levels according to their relative directions from the center. The position (i, j) belongs to the m-th level if $2\pi(m-1)/L_\theta \leq \theta_{i,j} < 2\pi m/L_\theta$, where $\theta_{i,j}$ is the counterclockwise angle from the vector $(0, 1)$ to the vector $(i - c_h, j - c_w)$. Combining the distance levels and the direction levels, the LRF is divided into $L_r \times L_\theta$ regions.

The LPSC kernel assigns a parameter to each region. All pixels of $\boldsymbol{X}$ falling into the same region share the same parameter. For the region with the l-th distance level and m-th direction level, the assigned parameter is denoted by $w_{l,m}$. The areas of regions increase with l, the farther away from the center, the larger the area, the more pixels sharing parameters. Because the center position of the kernel is important and forms the basis of regions, we assign an additional separate parameter $w_{0,0}$ for the center pixel. A conventional kernel with a size of $(2R+1) \times (2R+1)$ has $(2R+1)^2$ parameters, while a LPSC kernel only has $L_r \times L_\theta + 1$ parameters no matter how large R is. When R ranges from 2 to 9, a single conventional kernel has 25 to 361 parameters. In this range, it is sufficient to set L_r to 2 or 3 and set L_θ to 6 or 8, so an LPSC kernel only has 13 to 25 parameters.

Let $N_{l,m}$ denote the number of pixels falling into the region $bin(l, m)$ with the l-th distance level and the m-th direction level. In faraway regions with large l, $N_{l,m}$, the impacts of pixels in them should be weakened. Therefore, we regularize the weight $w_{l,m}$ of each region by $N_{l,m}$: $w_{l,m}/N_{l,m}$. As a result, the LPSC kernel aggregates finer information from pixels nearing the center and is less sensitive to those of pixels farther away. Similar to conventional convolution, the LPSC kernel is slide along the input feature map X with a pre-defined stride to perform convolution, as shown in Fig. 2(a). When the kernel is located at a spatial location (i, j), the output response is calculated as

$$(\boldsymbol{X} * \boldsymbol{W})(i, j) = \boldsymbol{W}(0, 0) \cdot \boldsymbol{X}(i, j) + \sum_{l=1}^{L_r} \sum_{n=1}^{L_\theta} \boldsymbol{W}(l, m) \cdot (\frac{1}{N_{l,m}} \sum_{u,v \in bin(l,m)} \boldsymbol{X}(u, v)) + b \quad . \quad (3)$$

For the LPSC kernel, the shape of its LRF is not necessarily a standard circle, but can be an oblique ellipse. As shown in Fig. 2(b), two additional hyper-parameters are introduced: the initial angle α and the eccentricity of the ellipse e. When dividing the regions, the distances are calculated according to the squared ellipse distance and the initial angle is added to the calculated directions. In this way, the LPSC kernel can better fit objects with different rotations and scales. In our experiments, we only evaluate the standard circular LRF by setting $\alpha = 0$ and $e = h/w = 1$.

3.2 LPSC VIA LOG-POLAR SPACE POOLING

Due to the special structure and parameter sharing, LPSC cannot be directly performed by popular deep learning frameworks. In this subsection, we show that LPSC can be readily implemented by conventional convolutions via log-polar space pooling to utilize efficient convolution modules.

Given the hyper-parameters R, L_r, L_θ, and g of the proposed LPSC, we can pre-compute a mask matrix $\boldsymbol{I}$ to indicate the region indexes of positions. The size of the mask $\boldsymbol{I}$ is $(2R+1) \times (2R+1)$. $1, \cdots, L_\theta \times L_r$ in $\boldsymbol{I}$ indicates the region index of the corresponding position. 0 indicates that the corresponding position does not fall into the LRF, since the region of the mask is the circumscribed rectangle of the LRF. The mask is slide through the input feature map $\boldsymbol{X}$ with the same stride of the LPSC convolution. As shown in Fig. 3(b), when the mask is located at a spatial location (i, j), pixels of $\boldsymbol{X}$ in the range are divided into regions indicated by the mask. All pixels in the same region are encoded into a single pixel by mean pooling. We re-arrange the pooled pixels of different regions into a matrix of $2L_r \times L_\theta/2$ to preserve their relative spatial positions, as shown in Fig. 3(a). In this way, given $H' \times W'$ convolution locations ($H' = H$ and $W' = W$ if the stride is 1 with padding), the spatial size of the output map $\boldsymbol{X}_p$ after log-polar space pooling equals $2H'L_r \times W'L_\theta/2$.

We perform conventional convolution with C' output channels on the output map $\boldsymbol{X}_p$ without padding. The size of the conventional convolution kernel is set to $(2L_r, L_\theta/2)$ and the stride is also $(2L_r, L_\theta/2)$. The output feature map $\boldsymbol{Y}_p$ has a size of $H' \times W' \times C'$. This is equivalent to performing the second term in Eq. (3). To model the first term, we use a separate 1×1 conventional convolution with the same C' channels on the original $\boldsymbol{X}$. The stride is the same as the log-polar space pooling. The output feature map $\boldsymbol{Y}_c$ contains the convolution responses of the center pixels. We add this separate center pixel convolution output $\boldsymbol{Y}_c$ to the contextual convolution output $\boldsymbol{Y}_p$. $\boldsymbol{Y}_c + \boldsymbol{Y}_p$ serves as the output feature map of the proposed LPSC.

3.3 INCORPORATING LPSC INTO DIFFERENT CNNS

LPSC can be integrated into different CNN architectures. A straightforward way is to replace all conventional convolution kernels with LPSC kernels in a part of convolution layers. For plain CNN architectures such as AlexNet (Krizhevsky et al., 2012) and VGGNet (Simonyan & Zisserman, 2014), we simply perform this strategy in lower layers to increase the LRFs. However, some network architectures such as ResNet (He et al., 2016) are constituted of specifically designed blocks. In ResNet, either the bottleneck or the basicblock structure only contains 3×3 and 1×1 convolutions. Due to the difference in the local receptive field, the information captured by these small convolutions and LPSC may be different. In order to better incorporate these two types of information, we propose a cross convolution strategy as an alternative to replacing all convolutions in each layer of the block. Specifically, we set a ratio p. For each of several consecutive layers, we replace $p\%$ of all convolution kernels to LPSC kernels, while the remaining $(100 - p)\%$ of conventional kernels remain the same. In this way, each convolution kernel in the next layer, whether it is a conventional or an LPSC kernel, perceives the outputs generated by both the conventional and LPSC kernels of the previous layer. We denote this cross-convolution strategy by LPSC-CC. Details on how to incorporate LPSCs depend on the CNN architecture and will be presented in Section 4.

3.4 DISCUSSIONS

Complexity. For a $(2M+1) \times (2N+1) \times C$ kernel, conventional convolution involves $(2M+1) \times (2N+1) \times C$ multiplications and $(2M+1) \times (2N+1) \times C$ additions. LPSC with L_r distance levels and L_θ direction levels only involves $2 * L_r \times L_\theta \times C$ multiplications, $(2M+1) \times (2N+1) \times C$ additions, and $(2M+1) \times (2N+1)$ lookups. The complexity of pre-computing the mask for lookup is $O(R^2)$, which only needs to be calculated once when initialing the layer. Typically, if $L_r = 2$, $L_\theta = 6$, LPSC only executes $24C$ multiplications for any size. However, even for a small $(2M+1) \times (2N+1) = 5 \times 5$ kernel, conventional convolution executes $25C$ multiplications; for a 9×9 kernel, multiplications increase to $81C$.

Structural benefits. With the special log-polar structure, the LPSC kernel naturally encodes the local spatial distribution of pixels w.r.t. the center and puts more attention to those adjacent pixels. Pixels with similar relative distances and directions share the same parameter, which not only reduces the number of parameters, but also makes the filter more robust and compact. Due to the

logarithm effect, when located at different objects, small objects are relatively enlarged, while large objects are relatively reduced. Therefore, LPSC is less sensitive to the size of objects.

Advantages of log-polar space pooling. In log-polar space pooling, we use mean pooling to achieve the regularization for regions. However, other pooling methods are also allowed. For example, if sum pooling is used, it is equivalent to remove the regularization. If max pooling is used, LPSC can be viewed as a special dilated convolution with irregular and data-driven holes.

Relation with effective receptive field (Luo et al., 2016). In Luo et al. (2016), it is found that the effective receptive field only occupies a fraction of the full theoretical receptive field. Therefore, convolutions with large LRF are required. It is also found that not all pixels in the LRF contribute equally, where the impacts of pixels near the center are much larger. The LPSC kernel follows this spirit to treat pixels near the center finely and increase the LRF exponentially.

Relation with advanced convolution methods. For the proposed LPSC, parameters distribute in the log-polar space. Most advanced convolution methods discussed in Section 2 do not change the regular sampling grid and focus on different aspects with LPSC. Therefore, these advances may be propagated to LPSC as well. For example, group and separable convolution (Chollet, 2017; Zhang et al., 2018) separates the channel dimension into groups and hence can be combined with the LPSC kernel in spatial dimensions. Graph convolution can also take advantage of LPSC as long as the relative distance and direction between nodes are defined.

Drawbacks. LPSC has two main drawbacks. (1) It introduces three additional hyper-parameters: L_r, L_θ, and g. However, in practice, their selectable ranges are quite limited. Generally, to make the 8-neighborhoods of the center pixel have finer and non-redundant regional resolution, L_r is set to 2 or 3, L_θ is set to 6 or 8, and g is set to 2 or 3. (2) Its implementation via log-polar space pooling incurs large memory overhead. The space complexity of the upsampled feature map $\boldsymbol{X}_p$ is $O(H'W'L_rL_\theta C)$. For a single layer, the space complexity of LPSC is $O(H'W'L_rL_\theta C + L_rL_\theta CC' + H'W'C')$.

4 EXPERIMENTS

We empirically study the effects and potential of the proposed LPSC on two tasks: image classification and semantic segmentation.

4.1 IMAGE CLASSIFICATION EXPERIMENTS

For image classification, we evaluate the behaviors of LPSC integrated with different CNN architectures on three datasets: CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and ImageNet (Russakovsky et al., 2015). We plug LPSC into three typical CNN architectures, including AlexNet (Krizhevsky et al., 2012), VGGNet-19 (Simonyan & Zisserman, 2014), and ResNet20 (He et al., 2016), by replacing a part of the conventional convolution layers. We use the Pytorch (Paszke et al., 2019) implementation[1] of these architectures as our baseline. For the AlexNet, there are 5 convolution layers each followed by a ReLU activation layer. The sizes of the convolution kernels are 11×11, 5×5, 3×3, 3×3, and 3×3, respectively. For the VGG19 Net, there are sixteen convolution layers. The kernel size for all convolution layers is 3×3. For the ResNet-20, there are 9 basic blocks. Each block contains two 3×3 convolution layers. A 3×3 convolution layer is applied before all blocks. When the conventional convolutions in a layer or block are replaced by LPSCs, the number of kernels and the size of the output feature map remain the same as the original convolution layer.

To make a fair comparison, all experimental setup and details including the learning rate, batch size, number of filters per layer, hyper-parameters for the optimizer (e.g., γ, momentum, weight decay) remain exactly the same as in the baseline. We did not tune any of these setups for our LPSC. Therefore, the differences in performances only come from the changes in convolution layers. The numbers of parameters are computed on the CIFAR-10 dataset. Top-1 accuracy is used as the performance measure.

Results on the CIFAR10 and CIFAR100 dataset. We train the AlexNet, VGGNet-19, and ResNet-20 with conventional convolution, dilation convolution, and LPSC five times by using different ran-

[1] https://github.com/bearpaw/pytorch-classification

Table 1: Comparison of different convolution methods.

(a) AlexNet

Convolution	Ori	Dilation	LPSC
# Params (M)	2.47	2.34	2.31
Acc. CIFAR-10 (%)	77.43 (0.25)	75.42 (0.06)	**78.44** (0.12)
Acc. CIFAR-100 (%)	43.98 (0.43)	44.43 (0.10)	**47.43** (0.20)

(b) VGGNet-19

Convolution	Ori	Dilation	LPSC
# Params (M)	20.04	20.08	20.08
Acc. CIFAR-10 (%)	93.54 (0.06)	93.46 (0.14)	**93.92** (0.06)
Acc. CIFAR-100 (%)	72.41 (0.17)	73.03 (0.34)	**73.13** (0.12)

(c) ResNet-20

Convolution	Ori	Dilation	LPSC	LPSC-CC
# Params (M)	0.27	0.27	0.27	0.27
Acc. CIFAR-10 (%)	91.66 (0.13)	91.44 (0.10)	**91.81** (0.21)	**92.01** (0.08)
Acc. CIFAR-100 (%)	67.56 (0.27)	66.90 (0.25)	**67.63** (0.27)	**68.09** (0.27)

dom seeds for initialization, respectively, and compare the average accuracies and standard deviations. "Mean accuracy (standard deviation)" results are reported in Table 1. We use LPSC in the first two convolution layers for AlexNet, in the added first convolution before all blocks for VGGNet-19, and in the first convolution layer before all residual blocks for ResNet-20. Hyper-parameters of the LPSC kernels in different layers and networks are the same as the first three columns in Table 7(d), respectively. These choices are based on the ablation study as described in the appendix A.1 and A.2. For dilation convolution, we replace the conventional convolutions with dilation convolution in the same layers in the three architectures, respectively, where the kernel size and dilation rates are set so that the LRF and number of parameters are comparable with LPSC. Specifically, for AlexNet, the kernel size and dilation rate are set to 5 and 2 in the first convolution layer, respectively, and 4 and 2 in the second convolution layer, respectively. For VGGNet-19, the kernel size and dilation rate are set to 4 and 2 in the added first convolution layer before all blocks, respectively. For ResNet-20, the kernel size and dilation rate are set to 4 and 3 in the first convolution layer before all residual blocks, respectively. These choices are based on the evaluations in Table 7 of the appendix A.2. From Table 1, we observe that LPSC outperforms dilation convolutions with comparable LRF and parameters. The standard deviations for LPSC are limited, which shows that LPSC is not particularly sensitive to initializations. In some cases, the worst results also exceed those of the original networks with conventional convolutions and dilation convolutions by a margin.

We also evaluate the cross convolution strategy for ResNet-20. We apply LPSC-CC to the layer before all blocks and all 3×3 layers of the first block with a fixed p of 50. We observe that the cross convolution strategy further improves the performances.

Comparison of FLOPs. On the CIFAR10 dataset with AlexNet, the FLOPs (recorded by the fvcore toolbox[2]) of conventional convolution, dilated convolution, and LPSC are 14.95M, 24.71M, and 11.42M, respectively. LPSC has much lower FLOPs than other convolution methods.

Results on the ImageNet dataset. ImageNet (Russakovsky et al., 2015) contains 1.28 million training images and 50k validation images from 1000 classes. We again use the Pytorch implementation[3] of ResNet-18 as the baseline. We replace conventional convolution with LPSC in the first convolution layer before all blocks of ResNet-18. Due to the limitation of computing resources, we reduced the batch size and learning rate by 4 times. Other hyper-parameters remain the same and we compare with the reported results in Tab. 2. LPSC slightly improves both top-1 and top-5 accuracies of ResNet-18 on this large-scale dataset.

[2]https://github.com/facebookresearch/fvcore

[3]https://github.com/bearpaw/pytorch-classification

Table 2: Results on the ImageNet dataset.

Model	ResNet-18	ResNet-18-LPSC
#params(M)	11.69	11.69
Top-1 Acc (%)	69.91	**70.09**
Top-5 Acc (%)	89.22	**89.26**

Table 3: Results on the VOC 2012 dataset. "-" means that the results are not reported. "*" indicates our reproduced results.

Method	oAcc	mAcc	fAcc	mIoU
DeepLabv3	-	-	-	0.701
DeepLabv3+	-	-	-	0.711
DeepLabv3+*	0.9230	0.8332	0.8652	0.7144
DeepLabv3+LPSC	**0.9273**	**0.8388**	**0.8714**	**0.7260**

4.2 SEGMENTATION ON PASCAL AND MEDICAL IMAGES

LPSC can also be applied to other vision tasks. On the PASCAL VOC 2012 dataset (Everingham et al., 2010; 2015) for general image semantic segmentation, we adopt the Pytorch implementation[4] of DeepLabv3+ (Chen et al., 2018b) with the MobileNet (Howard et al., 2017) backbone as the baseline. The training set is augmented by extra annotations provided in Hariharan et al. (2011). Overall accuracy (oAcc), mean accuracy (mAcc), freqw accuracy (fAcc), and mean IoU (mIoU) on the validation set are evaluated. In DeepLabv3+, the atrous spatial pyramid pooling (ASPP) module probes multi-scale features by applying atrous/dilated convolutions with three different rates. For DeepLabv3+LPSC, we replace the dilated convolution with the largest rate by LPSC in ASPP. The kernel size, L_r, L_θ, and g of LPSC are set to 9, 2, 8, 2, respectively. Comparisons with the reported and reproduced results are shown in Tab. 3. LPSC improves DeepLabv3+ by a margin of 1.1% on mIoU. All hyper-parameters and setups such as the learning rate, batch size, etc, remain the same, so the performance gains are only attributed to the proposed LPSC.

On the DRIVE dataset (Staal et al., 2004) for retinal vessel detection, we adopt CE-Net (Gu et al., 2019) as the baseline. Sensitivity (Sen), accuracy (Acc), and AUC are evaluated on the test set. The dense atrous convolution (DAC) block of CE-Net uses four cascade branches with increasing numbers of dilated convolutions. For CE-Net-LPSC-1, we replace the dilated convolutions with rates of 3 and 5 by LPSCs with sizes of 5 and 11 in DAC, respectively, so that LPSCs have the same LRFs with dilated convolutions. L_r, L_θ, and g of LPSCs are set to 2, 6, 3, respectively. For CE-Net-LPSC-2, we increase the kernel sizes of LPSCs to 9 and 15, respectively, to further increase LRFs. We accordingly use more parameters by setting L_r, L_θ, and g to 3, 8, 1.5, respectively. Other hyper-parameters remain the same[5]. We run our models ten times and report the average performances. Comparisons with the reported results are shown in Tab. 4. Our LPSC achieves good generalization performances on medical image segmentation with limited training samples.

[4]https://github.com/VainF/DeepLabV3Plus-Pytorch
[5]https://github.com/Guzaiwang/CE-Net

Table 4: Results on the DRIVE dataset (Staal et al., 2004). "*" indicates our reproduced results.

Method	Sen	Acc	AUC
HED (Xie & Tu, 2015)	0.7364	0.9434	0.9723
Azzopardi et al. (Azzopardi et al., 2015)	0.7655	0.9442	0.9614
Zhao et al. (Zhao et al., 2015)	0.7420	0.9540	0.8620
U-Net (Ronneberger et al., 2015)	0.7537	0.9531	0.9601
Deep Vessel (Fu et al., 2016)	0.7603	0.9523	0.9752
CE-Net (Gu et al., 2019)	0.8309	0.9545	0.9779
CE-Net* (Gu et al., 2019)	0.8266 (0.0106)	0.9550 (0.0009)	0.9782 (0.0008)
CE-Net-LPSC-1	0.8300 (0.0079)	**0.9552** (0.0011)	0.9782 (0.0008)
CE-Net-LPSC-2	**0.8312** (0.0075)	0.9548 (0.0011)	**0.9784** (0.0007)

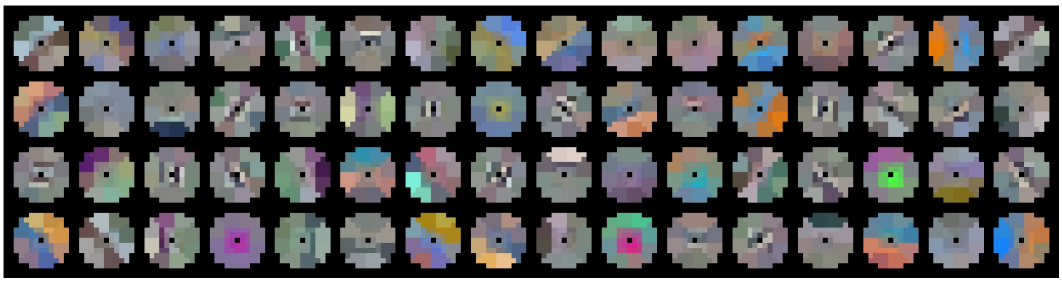
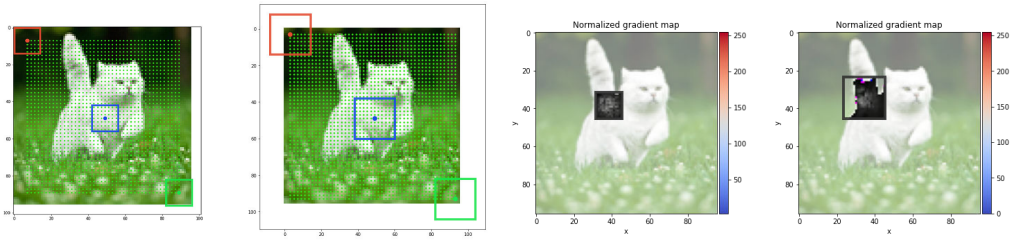


Figure 4: Visualization of the learned circular LPSC kernels without center convolution in the first convolution layer of Alexnet on the CIFAR-10 dataset.



(a) (b) (c) (d)

Figure 5: Estimated RFs with (a) conventional convolution and (b) LPSC. The normalized gradient map with (c) conventional convolution and (d) LPSC a sampled location.

4.3 VISUALIZATION

Visualization of the learned LPSC kernels. In Fig. 4, we visualize the learned LPSC kernels in the first convolution layer of AlexNet on the CIFAR-10 dataset. The 11×11 LPSC kernels have 3 distance levels and 8 direction levels. In LPSC kernels, the closer to the center, the higher the regional resolution; the more outward, the larger the range for parameter sharing. We observe that the learned LPSC kernels capture some special local structures and contextual configuration. In some kernels, the weights for adjacent regions are continuous; some kernels are sensitive to specific directions, edges, colors, or local changes; in some other kernels, specific combinations of regions are highlighted. More visualizations are shown in the appendix A.3.

Comparison of effective receptive field (ERF): Fig. 5(a) and (b) show the estimated RFs of SimpleVGGNet on the default example using conventional convolutions and LPSCs in the first two layers by the gradient-based RF estimation[6], respectively. LPSC enlarges the estimated RFs from 14×14 to 22×22. The normalized gradient maps w.r.t. a position of the output for estimating the RF using conventional convolutions and LPSCs are shown in Fig. 5(c) and (d). With LPSC, gradients can be back-propagated to more pixels of the input image.

5 CONCLUSION

In this paper, we have presented LPSC that naturally encodes the local contextual structures. LPSC distinguishes regions with different distance levels and direction levels, reduces the resolution of remote regions, and reduces the number of parameters by weight sharing for pixels in the same region. The LRF of LPSC increases exponentially with the number of distance levels. We impose a regularization on the parameters and implement LPSC with conventional convolutions by log-polar space pooling and separable center pixel convolution. We analyze the interests and drawbacks of LPSC from different aspects. We empirically show the effectiveness of the proposed LPSC on five datasets for classification and segmentation tasks.

[6]https://github.com/fornaxai/receptivefield

ETHICS STATEMENT

This paper proposes a new log-polar space convolution kernel and a method to achieve log-polar space convolution with conventional convolution. CNN has been applied to a wide range of applications. The proposed kernel can be applied to any CNN backbone to increase the local receptive field and reduce the number of parameters. Because this work presents such a general convolution module, we did not see any particular foreseeable ethics consequence.

REPRODUCIBILITY STATEMENT

We specify all the details on architectures, hyper-parameters, experimental settings, etc in Section 4, Appendix A.1 and Appendix A.2. Code is attached in the supplementary file.

REFERENCES

George Azzopardi, Nicola Strisciuglio, Mario Vento, and Nicolai Petkov. Trainable cosfire filters for vessel delineation with application to retinal images. *Medical image analysis*, 19(1):46–57, 2015.

Jonathan T Barron, Mark D Biggin, Pablo Arbelaez, David W Knowles, Soile VE Keranen, and Jitendra Malik. Volumetric semantic segmentation using pyramid context features. In *Proceedings of the IEEE international conference on computer vision*, pp. 3448–3455, 2013.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Advances in neural information processing systems*, pp. 831–837, 2001.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002.

Alexander C Berg and Jitendra Malik. Geometric blur for template matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2001.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 834–848, 2017a.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017b.

Liang-Chieh Chen, Maxwell D. Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jonathon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8713–8724, 2018a.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision*, pp. 801–818, 2018b.

Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proceedings of the IEEE international conference on computer vision*, pp. 3435–3444, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pp. 379–387, 2016.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. *International Conference on Learning Representations*, 2018.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Huazhu Fu, Yanwu Xu, Stephen Lin, Damon Wing Kee Wong, and Jiang Liu. Deepvessel: Retinal vessel segmentation via deep learning and conditional random field. In *International conference on medical image computing and computer-assisted intervention*, pp. 132–139. Springer, 2016.

Hang Gao, Xizhou Zhu, Steve Lin, and Jifeng Dai. Deformable kernels: Adapting effective receptive fields for object deformation. *arXiv preprint arXiv:1910.02940*, 2019.

Zaiwang Gu, Jun Cheng, Huazhu Fu, Kang Zhou, Huaying Hao, Yitian Zhao, Tianyang Zhang, Shenghua Gao, and Jiang Liu. Ce-net: Context encoder network for 2d medical image segmentation. *IEEE transactions on medical imaging*, 38(10):2281–2292, 2019.

Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *Proceedings of the IEEE international conference on computer vision*, pp. 991–998. IEEE, 2011.

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.

Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pp. 286–297. Springer, 1990.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Yunho Jeon and Junmo Kim. Active convolution: Learning the shape of convolution for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4201–4209, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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.

Duo Li, Anbang Yao, and Qifeng Chen. Psconv: Squeezing feature pyramid into one compact poly-scale convolutional layer. *arXiv preprint arXiv:2007.06191*, 2020.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 4898–4906, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Junran Peng, Ming Sun, Zhaoxiang Zhang, Tieniu Tan, and Junjie Yan. Pod: Practical object detection with scale-sensitive network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9607–9616, 2019.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Laura Sevilla-Lara, Deqing Sun, Varun Jampani, and Michael J Black. Optical flow with semantic segmentation and localized layers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3889–3898, 2016.

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

Joes Staal, Michael D Abràmoff, Meindert Niemeijer, Max A Viergever, and Bram Van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE transactions on medical imaging*, 23(4):501–509, 2004.

Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pp. 1451–1460. IEEE, 2018.

Tianyi Wu, Sheng Tang, Rui Zhang, Juan Cao, and Jintao Li. Tree-structured kronecker convolutional network for semantic segmentation. In *IEEE International Conference on Multimedia and Expo*, pp. 940–945. IEEE, 2019.

Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 1395–1403, 2015.

M. Yang, K. Yu, Z. Chi, Z. Li, and K. Yang. Denseaspp for semantic segmentation in street scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2528–2535. IEEE, 2010.

Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2018–2025. IEEE, 2011.

Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Scale-adaptive convolutions for scene parsing. In *Proceedings of the IEEE international conference on computer vision*, pp. 2031–2039, 2017a.

Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Perspective-adaptive convolutions for scene parsing. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):909–924, 2019.

Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions. In *Proceedings of the IEEE international conference on computer vision*, pp. 4373–4382, 2017b.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

Yitian Zhao, Lavdie Rada, Ke Chen, Simon P Harding, and Yalin Zheng. Automated vessel segmentation using infinite perimeter active contour model with hybrid region information with application to retinal images. *IEEE transactions on medical imaging*, 34(9):1797–1807, 2015.

A APPENDIX

A.1 ABLATION STUDY

Influence of hyper-parameters. The proposed LPSC kernel has four hyper-parameters: the kernel size $2R+1$, the number of distance levels L_r, the number of direction levels L_θ, and the growth rate g. For small kernels, since we set a minimum value 2 for the smallest distance level, the largest L_r can be determined by R^2 and g. If L_r is too large, no pixels will fall into regions of large distance levels. L_θ can be set to 6 or 8, since in most cases there are only about 8 pixels in the smallest circle in Fig. 1(b). We evaluate the influences of L_r, L_θ, and g by replacing the large 11×11 convolution kernels with the LPSC kernels in the first layer of AlexNet in Tab. 5(a) and by applying LPSC kernels with different sizes in the first convolution layer before all blocks of VGGNet and ResNet in Tab. 5(c) and (d), respectively. We run all models for only one time. Increasing the values of L_r and L_θ results in finer regions and improves the performances, but the number of parameters also increases.

VGGNet and ResNet use small 3×3 kernels. To make the number of parameters comparable, we fix (L_r, L_θ) to $(2, 6)$ and evaluate the influence of the kernel size $2R + 1$ in Tab. 5(c)(d). When $2R + 1$ is too small, the LRF is limited. When $2R + 1$ is too large, the regions with large distance levels may be coarse, i.e., a large amount of positions share the same weight, which may decrease the resolutions of parameters. Overall, for large kernels (11×11), we can set (L_r, L_θ, g) to $(3, 8, 2)$. For small kernels (5×5), we may fix (L_r, L_θ, g) to $(2, 6, 3)$[7].

Influence of the plugged layer. Tab. 5(b) and (d) show the results of using LPSC in different layers or blocks for AlexNet and ResNet-20, respectively. It seems that performing LPSC in low layers is more beneficial. This may be because pixels in high layers have merged information from large LRFs so that even adjacent pixels may have different influences on the center pixel and the weights for different positions are not suitable for sharing. Applying LPSC in low layers is conducive to increase the LRFs, filter redundant details, and back-propagate the gradients to more bottom pixels. In the last column of Tab. 5(d), the two successive convolution layers in each block are replaced with a single LPSC layer, so the number of convolution layers is reduced by half and the number of parameters is reduced by one third, but the performances are only reduced by 2% using half the layers with LPSC in ResNet.

Effects of weight regularization. In Tab. 6, we evaluate the effects of the weight regularization in Eq. (3) based on AlexNet. "Sum" shows the results by using sum pooling instead of mean pooling in log-polar space pooling in the first two LPSC layers. This is equivalent to remove the regularization and the performances are severely degraded. This is because far regions are exponentially larger than nearer regions. If positions in all regions are treated equally, even the weight for a far region is not too large, the accumulation of less relevant distant pixels will still produce an overwhelming response. We also try max pooling. It also performs worse than mean pooling. Due to the large LRF, regions with large distance levels for many adjacent center locations will have large overlaps.

[7] In this case, since we set a lower bound of the first level distance, $g = 2$ and 3 will result in the same LPSC kernel.

Table 5: Ablation study based on (a)(b) AlexNet (c) VGGNet-19 and (d) ResNet-20.

(a) 11×11 LPSC kernels with different hyper-parameters are used in the first convolution layer of AlexNet.

(L_r, L_θ, g)	(3,8,2)	(3,6,2)	(2,8,3)	(2,8,2)
# Params (M)	2.45	2.45	2.45	2.45
Acc. CIFAR-10 (%)	77.27	76.83	76.23	75.95
Acc. CIFAR-100 (%)	46.35	45.86	45.07	44.61

(b) Effects of using LPSCs in different convolution layers of AlexNet. Kernel size remains the same as in the corresponding layer. (L_r, L_θ, g) is fixed to $3, 8, 2$ and $2, 6, 3$ when applied to the 1st and 2nd layer, respectively.

LPSC layer	1	2	1 + 2
# Params (M)	2.45	2.33	2.31
Acc. CIFAR-10 (%)	77.27	**78.31**	78.28
Acc. CIFAR-100 (%)	46.35	44.81	**47.31**

(c) LPSCs with different sizes and hyper-parameters are used in an additionally added convolution layer before all blocks in VGGNet-19.

$2R+1$	5	9	13	17
(L_r, L_θ, g)	(2,6,3)	(2,6,3)	(2,6,3)	(3,8,4)
# Params (M)	20.08	20.08	20.08	20.08
Acc. CIFAR-10(%)	93.66	**94.01**	93.86	93.73
Acc. CIFAR-100(%)	72.95	73.13	73.08	**73.37**

(d) Effects of using LPSCs in different layers and blocks of ResNet-20. The first two rows denote the sizes and hyper-parameters of LPSCs in the first convolution layer before all blocks. The third row denotes the sizes of LPSCs with fixed hyper-parameters (2,6,2) in all blocks. "-" means that no LPSCs are used in blocks. "B" means that two successive 3×3 convolution layers are replaced with a single LPSC layer in the BasicBlock.

$2R+1$	5	9	13	5	13
(L_r, L_θ, g)	(2,6,2)	(2,6,3)	(2,6,3)	(2,6,2)	(2,6,2)
$2R+1$ for Blocks	-	-	-	5	9(B)
# Params (M)	0.27	0.27	0.27	0.39	0.18
Acc. CIFAR-10 (%)	91.55	91.67	**92.11**	91.82	89.78
Acc. CIFAR-100 (%)	67.23	67.19	66.97	**67.98**	65.10

Table 6: Effects of the weight regularization and center pixel convolution based on AlexNet.

Method	Sum	Max	No CenterConv	Mean
Acc. CIFAR-10(%)	21.61	76.65	**78.51**	78.28
Acc. CIFAR-100(%)	5.53	44.63	47.13	**47.31**

Some large responses may dominant repeatedly in many regions for different center locations, which suppress other useful local information.

Effects of center pixel convolution. In the fourth column of Tab. 6, we remove the center pixel convolution, i.e., the first term in Eq. (3). Center pixel convolution enlarges the importance of the center pixel. Contextual information itself may be sufficient for classification when there are few classes. For more complex tasks with more classes, center pixel convolution may provide complementary information.

Running times. On the CIFAR10 dataset, the training time for one epoch and the testing time for AlexNet are 3.4808 and 0.9083, respectively; after replacing conventional convolutions with LPSCs in the first two layers, the training time for one epoch and the testing time are 23.3981 and 3.7476, respectively. In our implementation, LPSC runs much slower than conventional convolution, but this

Table 7: Comparison with different convolution methods based on different architectures.

(a) AlexNet

Conv Type	Ori	Dilation (size, dilation rate)					Deformable				Square (size, pool size)		
hy.-para. L-1	-	5,3	7,2	-	-	7,2					11,5	-	11,5
hy.-para. oth.		-	-	3,2	3,3	3,3					-	9,3	9,3
Layer	-	1	1	2	2	1,2	1	2	1,2	all	1	2	1,2
#params(M)	2.47	2.45	2.46	2.28	2.28	2.26	2.47	2.55	3.21	7.04	2.45	2.28	2.26
CIFAR10	77.22	73.43	75.7	74.94	78.11	75.95	75.98	55.34	32.96	55.22	76.10	75.17	73.26
CIFAR100	43.87	44.86	45.98	41.08	44.21	44.26	41.96	30.30	30.36	30.82	44.50	42.43	41.89

(b) VGGNet-19

Conv Type	Ori	Dilation (size, dilation rate)				Deformable				Square (size, pool size)		
hy.-para. L-1+	-	-	-	-	-					13,4	9,3	9,3
hy.-para. oth.		3,2	3,2	3,2	3,2					-	-	5,3
Block	-	1	2	1,2	1,2,3	1+	1+,1	1+,1,2	1+,all	1+	1+	1+,1
#params(M)	20.04	20.04	20.04	20.04	20.04	20.04	20.11	20.48	58.53	20.04	20.04	20.04
CIFAR10	93.34	91.53	91.97	89.61	89.56	92.53	70.37	90.64	90.02	87.42	90.01	89.4
CIFAR100	71.95	68.46	69.30	63.75	63.83	69.32	37.50	64.37	61.26	62.51	66.64	65.79

(c) ResNet-20

Conv Type	Ori	Dilation (size, dilation rate)				Deformable		Square (size, pool size)			
hy.-para. L-1+	-	3,2	3,3	5,2	3,2			13,4	9,3	9,3	9,3
hy.-para. oth.		-	-	-	3,2			-	-	5,3	5,3
Block	Ori	1+	1+	1+	1+,1,2,3	1+	1,2,3	1+	1+	1+,1	1+,1,2
#params(M)	0.27	0.27	0.27	0.27	0.27	0.27	0.78	0.27	0.27	0.27	0.28
CIFAR-10	91.96	91.39	91.77	91.33	86.09	90.27	46.57	88.09	89.84	88.43	87.19
CIFAR-100	67.83	66.95	67.58	67.08	59.26	65.44	13.45	61.43	63.54	62.05	60.37

(d) Results of LPSC with different architectures

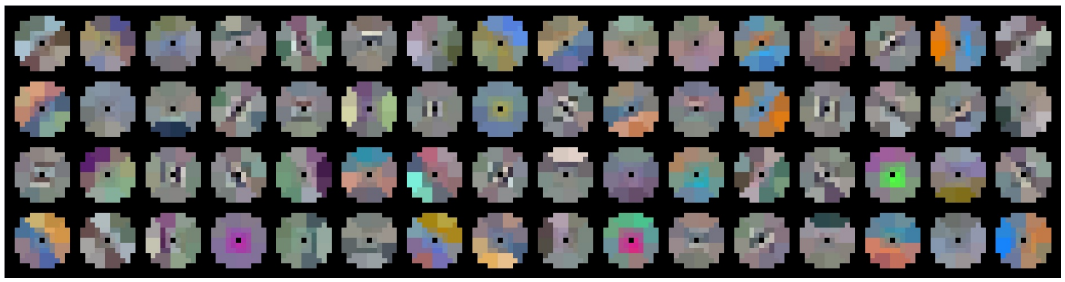
Architecture	AlexNet	VGGNet	ResNet $(size, L_r, L_\theta, g)$	
hy.-para. L-1+	11,3,8,2	9,2,6,3	13,2,6,3	5,2,6,2
hy.-para. oth.	9,2,6,3	-	-	5,2,6,2
Layer/Block	1,2	1+	1+	1+,all
#params(M)	2.31	20.08	0.27	0.39
CIFAR-10	**78.28**	**94.01**	**92.11**	91.82
CIFAR-100	**47.31**	**73.13**	66.97	**67.98**

is because we use of-the-shell conventional convolution modules to implement LPSC. To this end, we must first apply log-polar space pooling with the fold and unfold operations in Pytorch, which consume much time and space complexity. LPSC can be greatly accelerated if it is directly implemented with CUDA or by directly adapting the underlying code of convolutions in the integrated framework.

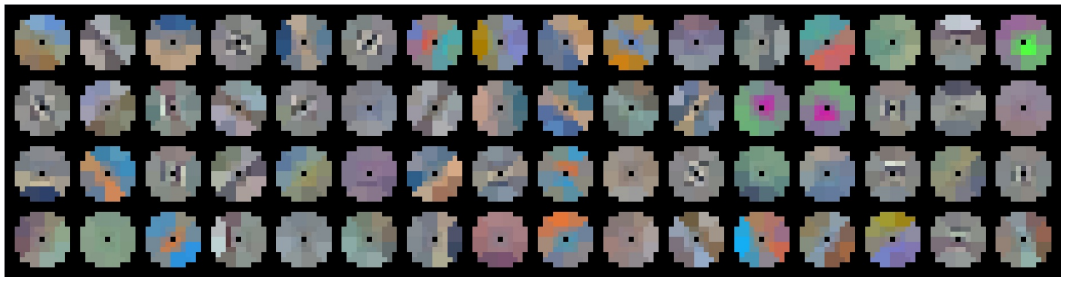
A.2 Comparison with other convolution methods

Dilated convolution (Chen et al., 2017b) can also exponentially increase the LRF without increasing the number of parameters. Deformable convolution (Dai et al., 2017) adaptively adjusts the LRF by using additional parameters to infer the offsets for each position. We use the Pytorch implementation of deformable convolution[8] in our experiments. Different from LPSC where regions are divided in the log-polar space, we can also averagely divide the kernel into different square regions, e.g., a 9×9 kernel can be divided into 3×3 regions with a size of 3×3. All positions in the same square region share the same parameter. We denote this alternative convolution method by *square*

[8]https://github.com/oeway/pytorch-deform-conv



(a)



(b)

Figure 6: Visualization of the learned circular LPSC kernels without center convolution in the first convolution layer of Alexnet on (a) the CIFAR-10 dataset and (b) the CIFAR-100 dataset.

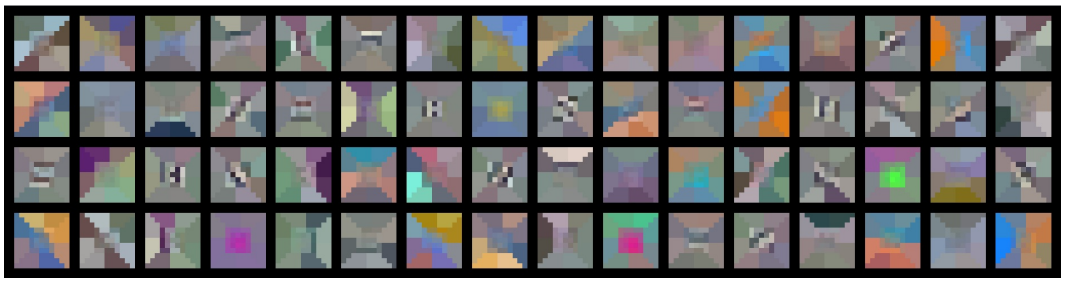
convolution, which also increases LRF with fewer parameters. Similarly, they can also be used to replace conventional convolution at different layers in different architectures.

In Tab. 7, we compare LPSC with conventional convolution, dilation convolution, deformable convolution, and square convolution in the three architectures, respectively. For dilation convolution, square convolution, and LPSC, the second and third rows show the hyper-parameters of the corresponding kernels in the first convolution layer and in other layers or blocks, respectively. We run all models for only one time. “Size” indicates the kernel size. For VGGNet and ResNet, “1+” indicates the first convolution layer before all blocks. The fourth row indicates the indexes of layers or blocks in which conventional convolution is replaced by the corresponding convolution.

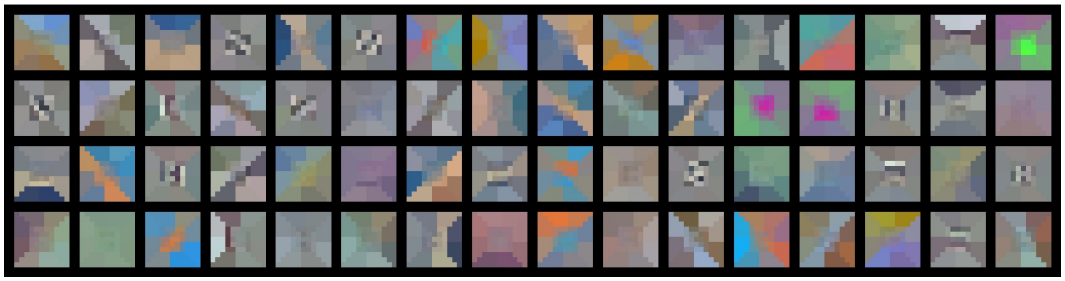
The second columns show the results of the three original architectures. With fewer or comparable parameters, LPSC outperforms conventional convolution in AlexNet and VGGNet, and obtains comparable results in ResNet.

The third column blocks show the results of dilated convolution with different hyper-parameters in different layers. The hyper-parameters, including the kernel size and the dilation rate, are set to keep the total number of parameters and the LRF comparable to conventional convolution and LPSC. In AlexNet and VGGNet, LPSC outperforms dilated convolution with different hyper-parameters significantly. This shows the effectiveness of spatial structure and parameter sharing in LSPC. Applying dilated convolution in higher layers also leads to worse performances, which further verifies our analysis in Sec. A.1, but LPSC outperforms dilated convolution. When only used in the first layer of ResNet, sometimes dilated convolution achieves slightly better results than LPSC. The reason may be that ResNet can stack deeper layers with residual connections and hence achieve large LRF using multi-layer small regular kernels, which eases the need for large LRF in lower layers. Moreover, the non-uniform distribution of parameters in LPSC may cause information dispersion and over-smoothing, therefore it is more difficult to model residuals.

Deformable convolution introduces additional parameters. When applied to all blocks of VGGNet or ResNet, the parameters are more than doubled. Deformable convolution causes performance degradation of different architectures. LPSC also outperforms the average square convolution with different hyper-parameters. This indicates that the spatial structure designed in log-polar space can better capture the contextual information.

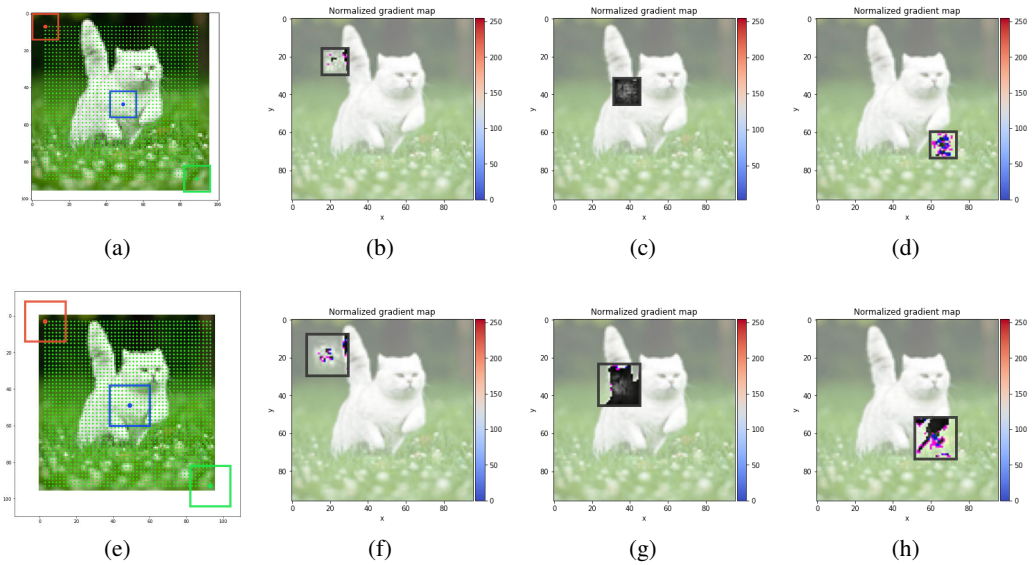


(a)



(b)

Figure 7: Visualization of the filled 11×11 LPSC kernels without center convolution in the first convolution layer of Alexnet on (a) the CIFAR-10 dataset and (b) the CIFAR-100 dataset.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 8: Estimated RFs with (a) conventional convolution and (e) LPSC. The normalized gradient map with (b-d) conventional convolution and (f-h) LPSC at different locations.

A.3 VISUALIZATION

Visualization of the learned LPSC kernels. In Fig. 6, we visualize the learned LPSC kernels in the first convolution layer of AlexNet on the CIFAR-10 and CIFAR-100 datasets. The LPSC kernels have a size of 11×11, 3 distance levels, 8 direction levels, and a growth factor of 2. Different from conventional convolution kernels, in LPSC kernels, the closer to the center, the higher the regional resolution; the more outward, the larger the range for parameter sharing. We observe that the learned LPSC kernels capture some special local structures and contextual configuration. In

some kernels, the weights for adjacent regions are continuous; some kernels are sensitive to specific directions, edges, colors, or local changes; in some other kernels, specific combinations of regions are highlighted. To fully utilize the space circumscribed by the LRF of the kernel, we fill four corners (positions that do not fall into the LRF) with the weights of corresponding nearest regions in all experiments, respectively, as shown in Fig. 7.

Comparison of effective receptive field (ERF): Fig. 8(a) and (e) show the estimated RFs of SimpleVGGNet on the default example using conventional convolutions and LPSCs in the first two layers by the gradient-based RF estimation[9], respectively. LPSC enlarges the estimated RFs from 14×14 to 22×22. The normalized gradient maps w.r.t. different positions of the output for estimating the RFs using conventional convolutions and LPSCs are shown in Fig. 8(b-d) and Fig. 8(f-h), respectively. With LPSC, gradients can be back-propagated to more pixels of the input image.

[9]https://github.com/fornaxai/receptivefield