SMPConv: Self-moving Point Representations for Continuous Convolution

Sanghyeon Kim[1] Eunbyung Park[1,2*]
[1]Department of Electrical and Computer Engineering, Sungkyunkwan University
[2]Department of Artificial Intelligence, Sungkyunkwan University

Abstract

Continuous convolution has recently gained prominence due to its ability to handle irregularly sampled data and model long-term dependency. Also, the promising experimental results of using large convolutional kernels have catalyzed the development of continuous convolution since they can construct large kernels very efficiently. Leveraging neural networks, more specifically multilayer perceptrons (MLPs), is by far the most prevalent approach to implementing continuous convolution. However, there are a few drawbacks, such as high computational costs, complex hyperparameter tuning, and limited descriptive power of filters. This paper suggests an alternative approach to building a continuous convolution without neural networks, resulting in more computationally efficient and improved performance. We present self-moving point representations where weight parameters freely move, and interpolation schemes are used to implement continuous functions. When applied to construct convolutional kernels, the experimental results have shown improved performance with drop-in replacement in the existing frameworks. Due to its lightweight structure, we are first to demonstrate the effectiveness of continuous convolution in a large-scale setting, e.g., ImageNet, presenting the improvements over the prior arts. Our code is available on https://github.com/sangnekim/SMPConv

1. Introduction

There has been a recent surge of interest in representing the convolutional kernel as a function over a continuous input domain. It can easily handle irregularly sampled data both in time [1, 59] and space [61, 65], overcoming the drawbacks of the discrete convolution operating only on discretized sampled data with pre-defined resolutions and grids. With the progress in modeling and training continuous kernels, it has enjoyed great success in many practical scenarios, such as 3D point cloud classification and segmentation [36,41,52,58,64], image super resolution [57], object tracking [10], to name a few. Furthermore, the recent trends of using large convolutional kernels with strong empirical results urge us to develop a more efficient way to implement it [13,32], and the continuous convolution will be a promising candidate because of its capability to readily construct arbitrarily large receptive fields [44,45].

One of the dominant approaches to modeling the continuous kernel is to use a particular type of neural network architecture, taking as inputs low-dimensional input coordinates and generating the kernel values [44,45], known as neural fields [38,49] or simply MLPs. Using neural fields to represent the kernels, we can query kernel values at arbitrary resolutions in parallel and construct the large kernels with a fixed parameter budget, as opposed to the conventional discrete convolutions requiring more parameters to enlarge receptive fields. Thanks to recent advances to overcome the spectral bias on training neural fields [49], they can also represent functions with high-frequency components, which enables learned kernels to capture fine details of input data.

While promising in various tasks and applications, this approach has a few downsides. First, it incurs considerable computational burdens to already computation-heavy processes of training deep neural networks. Each training iteration involves multiple forward and backward passes of MLPs to generate kernels and update the parameters of MLPs. This additional complexity prevents it from being applied to large-scale problems, such as ImageNet-scale, since it needs deeper and wider MLP architectures to construct more complex kernels with more input and output channels. Although MLPs can generate larger sizes and numbers of kernels without adding more parameters, it has been known that the size of MLPs mainly determines the complexity of the functions they represent and, eventually, the performance of the CNNs.

Furthermore, the kernels generated by an MLP depend heavily on the architectural priors. As a universal function approximator, a neural network with sufficient depth and width can express any continuous functions [25]. However, we have empirically observed strong evidence that the ar-

*Corresponding authors

chitecture of neural networks has played a significant role in many practical settings, suggesting that various architectural changes to the kernel-generating MLPs would significantly affect the performance of CNNs. Considering a large number of hyperparameters in training neural networks, adding more knobs to tune, e.g., activation functions, width, depth, and many architectural variations of MLPs., would not be a pragmatic solution for both machine learning researchers and practitioners.

In this paper, we aim to build continuous convolution kernels with negligible computational cost and minimal architectural priors. We propose to use moving point representations and implement infinite resolution by interpolating nearby moving points at arbitrary query locations. The moving points are the actual kernel parameters, and we connect the neighboring points to build continuous kernels. Recent techniques in neural fields literature inspire it, where they used grid or irregularly distributed points to represent features or quantities in questions (density or colors) for novel view synthesis [6, 50, 63, 66, 67]. The suggested approach only introduces minor computational costs (interpolation costs) and does not consist of neural networks (only point representations and interpolation kernels). Moreover, the spectral bias presented in training MLPs [43] does not exist in the suggested representation. Each point representation covers the local area of the input domain and is updated independently of each other, contrasted with MLPs, where updating each parameter would affect the entire input domain. Therefore, highly different values of nearby points can easily express high-frequency components of the function.

The proposed method can also be more parameter efficient than the discrete convolution to construct the large kernels. Depending on the complexity of the kernels to be learned, a few numbers of points may be sufficient to cover a large receptive field (e.g., a unimodal function can be approximated by a single point). Many works have extensively exploited non-full ranks of the learned kernels to implement efficient convolutions or compress the models [32]. Our approach can likewise benefit from the presence of learned kernels with low-frequency components.

We conduct comprehensive experimental results to show the effectiveness of the proposed method. First, we demonstrate that moving point representations can approximate continuous functions very well in 1D and 2D function fitting experiments. Then, we test its ability to handle long-term dependencies on various sequential datasets. Finally, we also evaluate our model on 2D image data. Especially, we perform large-scale experiments on the ImageNet classification dataset (to our knowledge, it is the first attempt to use continuous convolution for such a large-scale problem). Our proposed method can be used as a drop-in replacement of convolution layers for all the above tasks without bells and whistles. The experimental results show that it consistently improves the performance of the prior arts.

2. Related works

Neural fields and continuous convolution. Neural fields have recently emerged as an alternative neural network representation [49]. It is a field parameterized by a neural network, such as simple MLP, taking as low-dimensional coordinates input and generating quantity in questions. It has shown great success in various visual computing domains, such as novel view synthesis [38], 3D shape representation [37, 40], and data compression [17], to name a few. Since it is a function over a continuous input domain, it can produce outputs at arbitrary resolutions. Recent studies have exploited its continuous nature to model continuous kernels for CNNs [44, 45, 58]. [58] used an MLP architecture to implement the continuous kernel to handle irregularly sampled data, such as 3D point clouds. While successful, the descriptive power of the learned kernel is limited due to its bias toward learning low-frequency components. With the recent advances in overcoming the spectral bias [49], [45] has explored various activation functions to improve the performance of CNNs. To further improve, [44] proposed to learn the receptive field sizes and showed impressive performance on multiple tasks.

We also propose a method that can likewise be classified as a neural field. However, we implement a field without neural networks, using self-moving point representations and interpolation schemes. It significantly reduces the computational costs to compute the kernels during training compared to the conventional MLP-based methods. Furthermore, it removes the burden of numerous hyperparameter searches for a newly introduced neural network in the already complicated training procedure.

Grid and point representations. Although MLP-based neural fields have succeeded in many tasks, they require substantial computational costs for training and inference, and the spectral bias presented in MLPs often degrades the performance. In order to reduce the computations and avoid the issues of using MLPs, classical grid-based representations have been adopted in the neural fields. Plenoxels [66] stores the coefficients of spherical harmonic in the 3D voxel structure and implements the infinite resolution via the interpolation methods, reducing the training time from days to minutes. Instant-NGP [39] further optimized the speed using a hash-based indexing algorithm. The combination of grid representations and MLPs has been extensively explored to find better solutions [6, 50].

Point representations have been recently suggested in neural fields to improve the shortcomings in grid-based representations, thanks to their flexibility and expressibility [63, 67]. They can generate outputs over a continuous input domain by interpolating neighboring points given a

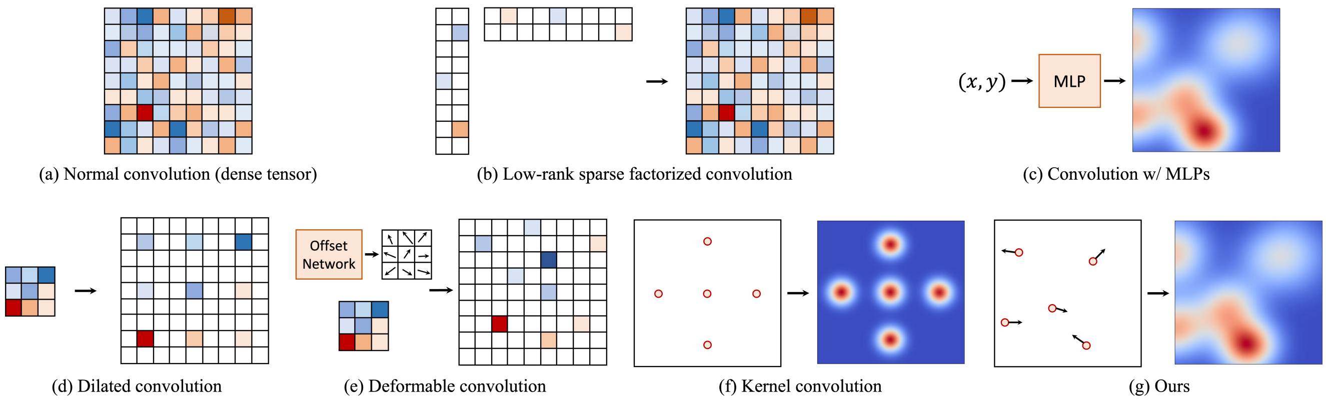

Figure 1. A various methods for large kernel construction.

query point. It can be more accurate and parameter efficient than the grid-based ones since it can adaptively locate points, considering the complexity of functions, e.g., fewer points in low-frequency areas. Although our work is largely inspired by these works, 1. we repurpose the point representations to build continuous convolutional kernels, 2. we do not use MLPs that would introduce additional modeling complexity, 3. our method does not require complex initializations (or algorithms) to relocate points, such as depth-based initialization [63] and mask-based initialization [67].

Large kernel convolution. Since the great success of VGG-style CNNs [24, 26, 48, 51] in ImageNet [12], we repeatedly apply small kernels (e.g., 3×3 kernel) with deep layers to get a large receptive field and deal with long-term dependencies. With the massive success of transformers in the vision domain [15, 16, 34, 55], which have a large receptive field, recent studies have started to revisit the descriptive power of large kernels [13, 32, 35, 53]. RepLKNet [13] has shown we can increase the kernel size up to 31×31 with the help of the reparameterization trick. More recently, SLaK [32] managed to scale the kernel size to 51×51 and 61×61, showing improved performance with dynamic sparsity and low-rank factorization (Fig. 1-(b)). While promising, the number of parameters increases according to the kernel size, which is a major bottleneck in representing large kernels.

The continuous kernel can construct large kernels with a fixed parameter budget (Fig. 1-(c)). CKConv [45] and FlexConv [44] have exploited this property and demonstrated that their method can model long-term dependencies by constructing large kernels on various datasets. However, they introduce a considerable amount of computations to the training process, and they have yet to perform large-scale experiments, e.g., ImageNet. To the best of our knowledge, our approach is the first to conduct such a large-scale experiment using continuous convolution.

On the other hand, dilated convolutions [7, 8] can also be used to enlarge receptive fields with a small number of parameters, and they also do not introduce additional computations during training (Fig. 1-(d)). Deformable convolutions [9] look similar to ours in terms of moving points arbitrarily (Fig. 1-(e)). However, they learn how to deform the kernels (or predict the offset) during inference. On the other hand, we adjust the locations of the point representations during training to find the optimal large kernels. A concurrent work suggests learning the offsets during training [23]. In contrast to ours, it is a discrete formulation, thus losing the benefits of continuous convolution. Furthermore, we adjust the receptive fields of each point representation separately, yielding more expressive representations.

Continuous convolution for point clouds. There have been many continuous convolution approaches to handle 3D point cloud data, which is an important example of irregularly sampled data. PointNet [41] and PointNet++ [42] are pioneer works that use average pooling and 1×1 convolution to aggregate features. [31, 33, 58, 60] leveraged MLPs to implement continuous convolution. KPConv [52] is also considered a continuous convolution and shares some similarities with ours (Fig. 1-f). They also used point representation and interpolation kernels for handling point clouds. However, their points are fixed over the training, unlike ours. They also proposed a deformable version, which requires additional neural networks to predict the offset of the kernel points.

3. SMPConv

3.1. Self-moving point representation

This section describes the proposed self-moving point representation to represent a continuous function. Let d be the size of the input coordinates dimension, e.g., 1 in time-series data and 2 in the spatial domain. $\texttt{SMP} : \mathbb{R}^d \to \mathbb{R}^{N_c}$ is a vector-valued function, mapping from the input coordinates to the output kernel vectors, where N_c is a channel size. Given a query point $x \in \mathbb{R}^d$, we define a continuous kernel function as follows.

$$\text{SMP}(x;\phi) = \frac{1}{|\mathcal{N}(x)|} \sum_{i \in \mathcal{N}(x)} g(x, p_i, r_i) w_i, \qquad (1)$$

where $\phi = \{\{p_i\}_{i=1}^{N_p}, \{w_i\}_{i=1}^{N_p}, \{r_i\}_{i=1}^{N_p}\}$ is a set of learnable parameters, and N_p is the number of points that are used to represent the function. $p_i \in \mathbb{R}^d$ is the coordinates of self-moving point representation $w_i \in \mathbb{R}^{N_c}$, and each point representation also has a learnable radius, $r_i \in \mathbb{R}^+$ is a positive real number, which we implement it by clipping for numerical stability. We define a distance function $g : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^+ \rightarrow \mathbb{R}$ as follows,

$$g(x, p_i, r_i) = 1 - \frac{\|x - p_i\|_1}{r_i}, \qquad (2)$$

where $\|\cdot\|_1$ is a L1 distance. $\mathcal{N}(x)$ is a set of indices of neighboring points of a query coordinate x, defined as $\mathcal{N}(x) = \{i \,|\, g(x, p_i, r_i) > 0, \forall i\}$. Thus, points beyond a certain distance (depending on the radius) will not affect the query point. Hence, SMP generates output vectors by a weighted average of the nearby point representations. Note that all three parameters $\{p_i\}, \{w_i\}, \{r_i\}$ are jointly trained with the CNN model parameters, and the gradients w.r.t those parameters can be easily computed using an automatic differentiation library. As the name SMP suggests, the coordinates $\{p_i\}$ are updated during training, resulting in moving points representation.

Compared to a fixed-point representation, where $\{p_i\}$ are not trainable, ours can approximate complex functions more precisely. Since each point can move freely, more points can be gathered in high-frequency areas. On the other hand, few points can easily represent low-frequency components, resulting in more parameter-efficient representation. For example, a single point may be sufficient to approximate unimodal functions.

3.2. SMPConv

We leverage the suggested representation to implement a continuous convolution operator. In one dimensional case, $d = 1$, a continuous convolution can be formulated as,

$$(f * \text{SMP})(x) = \sum_{c=1}^{N_c} \int_{\mathbb{R}} f_c(\tau) \text{SMP}_c(x - \tau) d\tau, \qquad (3)$$

where $f : \mathbb{R} \rightarrow \mathbb{R}^{N_c}$ is a input function and the f_c and SMP_c denote the c-th element of the inputs. The convolution operator generates a function, computing the filter responses by summing over entire N_c channels. One SMP representation corresponds a convolution operator, and multiple SMPs are used to implement one convolutional layer to generate multiple output channels. In contrast to the previous MLP-based continuous convolution, which uses one neural network for one convolutional layer, our approach has separate

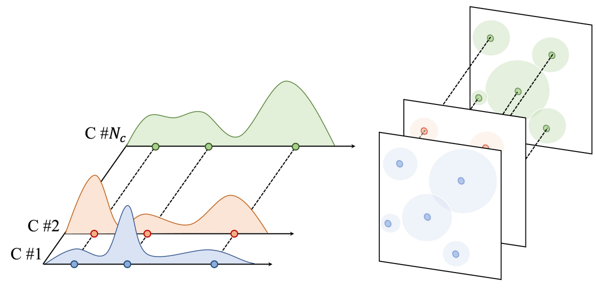

Figure 2. Self-moving point representation. (a) SMP as a function of the one-dimensional input domain, and (b) SMP as a function of the two-dimensional input domain. 'C #1' means the first channel. Each channel shares the location of the points, whereas each channel has its own weight parameters.

parameters for each convolution filter in a layer. It gives more freedom to each filter and results in more descriptive power of the learned filter.

As depicted in Fig. 2, the kernels of each filter share the position parameters. That is, each filter of one layer has its own position parameters. Although we could use different SMP for different channels. However, it will considerably increase the number of learnable parameters ($\{p_i\}$ per channel), and we believe that locating points at the same location for a convolutional filter can be a reasonable prior, where a convolutional filter can focus on specific areas in the input domain.

We leverage our continuous formulation to construct large kernels, motivated by the recent success of using them in many tasks. We can create arbitrary size large kernels by querying multiple discretized coordinates to SMP.

3.3. Training

Training a large kernel has been challenging and computationally heavy, and naive training practice has yet to show promising results. Recently, [13, 14] proposed a reparameterization trick to combine different-size kernels as a separate branch, resulting in improved performance and more stable training. We also applied the same trick to train CNNs with SMPConv.

We empirically figured out that performance degradation occurs when the coordinates $\{p_i\}$ are forced to fit inside the kernel by clipping. Thus, we let the coordinates be freely updated during training.

We also found that the initialization of the parameters ϕ matters. For point locations, $\{p_i\}$, we randomly sample from a gaussian distribution with small σ. It initially locates the points in the center and gradually spreads out over the training process. We empirically found that this strategy yields more stable training, especially at the beginning of the training. We also initialized with small values

Method	k	Params.	Time ↓	Throughput ↑
Deformable [9]	3	0.29M	61.2	**4390.7**
	5	1.37M	157.3	1618.9
	7	4.39M	293.1	882.3
FlexConv [44]	33	0.67M	92.9	1923.4
SMPConv	33	0.49M	**40.1**	4258.4

Table 1. Training time (sec/epoch) and throughput (examples/sec) comparison with CIFAR10 on a single RTX3090 GPU. Both are tested with a batch size of 64 and input resolution of 32 × 32. The k is kernel size. The time is the average training time of the first 10 epochs.

for $\{r_i\}$, which each weight parameter firstly has a narrow sight. Over the course of training, $\{r_i\}$ also gradually increases if necessary.

3.4. Efficiency

Assuming the size of a convolution filter is $C \times N \times N$, where C is the number of kernels and N is the height and width of filter, CN^2 parameters are required in dense convolution (Fig. 1-(a)). Therefore, the number of parameters is proportional to kernel resolution $N \times N$. On the other hand, SMP needs $(1 + d + C)N_p$ parameters, where d is the size of the input coordinates dimension, and N_p is the number of weight points. We used $N_p \ll N^2$, so SMP is more efficient than dense convolution in terms of the number of parameters. Furthermore, as the number of parameters does not depend on kernel resolution, SMP can represent kernels of any size, such as large or continuous kernels with fixed budget parameters.

Due to the point representations and interpolation schemes without supplementary neural networks, SMPConv has an advantage of computational complexity. On the other hand, the existing large kernel convolutions are computationally heavy. Deformable convolution (Fig. 1-(e)), for example, requires offset prediction networks and convolution with interpolated inputs *during both training and inference*, resulting in additional computation and parameter costs. Additionally, it relies on dense convolution, making it impractical to increase kernel size significantly. Similarly, MLP-based methods(Fig. 1-(c)) like FlexConv [44] leverage kernel generation neural network, and it also increases computational burdens. The results presented in Tab. 1 demonstrate that SMPConv outperforms the existing large kernel convolutions in terms of speed.

4. Experiments

4.1. Continuous function approximation

Firstly, we conducted a fitting experiment to validate that our self-moving point representation can work as an approximator for a continuous function. To do so, we used

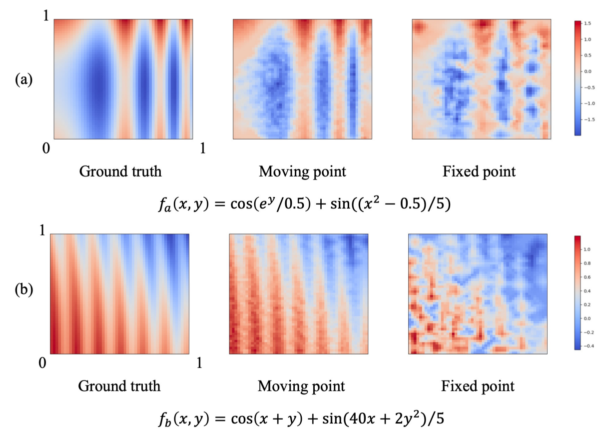

Figure 3. Comparison between the moving point and the fixed point representations through the fitting. Our proposed moving point representations can approximate given continuous functions with higher accuracy.

two sinusoidal-based functions as the ground truth. Given a function, SMP is optimized to represent the sampled function on a 51 × 51 grid. In this experiment, we designed SMP with 204 points. The fitting result has been shown in Fig. 3. It demonstrates that our proposed method reasonably well approximates a given continuous function with fewer points. Additionally, we compared with the fixed point representation and observed that optimizing the position of points together helps to better approximate the function as the number of points is equal.

4.2. Sequential data classification

To demonstrate that SMPConv can handle long-term dependencies well, we evaluated our method on various sequential data tasks, such as sequential image and time-series classification. To do so, we followed FlexTCN [44] to construct a SMPConv architecture for causal 1D CNN whose kernel size is same as the input sequence length. We substituted their parameterized kernels with ours without additional modifications. To maintain a similar number of network parameters, SMPConv contains 30 weight points in each SMP. To alleviate the computation burdens caused by the convolution with large kernels, we have considered exploiting the computations through a fast Fourier transform. More network and experimental details are in Appx. 1.1.

Sequential image. We tested our SMPConv on the 1D version of images from the datasets, sequential MNIST (sMNIST), permuted MNIST(pMNIST) [29], and sequential CIFAR10 (sCIFAR10) [4]. These datasets have long input sequence lengths, for example, 784 for sMNIST and pMNIST, and 1024 for sCIFAR10. Note that it is hard to model these datasets without proper kernel representations. As shown in Tab. 2, the proposed model has achieved state-

Model	Params.	sMNIST	pMNIST	sCIFAR10
DilRNN [4]	44k	98.0	96.1	-
LSTM [2]	70k	87.2	85.7	-
GRU [2]	70k	96.2	87.3	-
TCN [2]	70k	99.0	97.2	-
r-LSTM [56]	500k	98.4	95.2	72.2
IndRNN [30]	83k	99.0	96.0	-
TrellisNet [3]	8M	99.20	98.13	73.42
URLSTM [20]	-	99.28	96.96	71.00
HiPPO [18]	0.5M	-	98.30	-
coRNN [47]	134k	99.4	97.3	-
CKCNN [45]	98k	99.31	98.00	62.25
LSSL [21]	-	99.53	98.76	84.65
S4 [19]	-	99.63	98.70	**91.13**
FlexTCN [44]	375k	99.62	98.63	80.82
Ours	373k	**99.75**	**99.10**	**84.86**

Table 2. Sequential image classification results.

Model	Params.	CT	SC	SC-raw
GRU-ODE [11]	89k	96.2	44.8	$\sim$ 10.0
GRU-Δt [27]	89k	97.8	20.0	$\sim$ 10.0
GRU-D [5]	89k	95.9	23.9	$\sim$ 10.0
ODE-RNN [46]	89k	97.1	93.2	$\sim$ 10.0
NCDE [27]	89k	98.8	88.5	$\sim$ 10.0
CKCNN [45]	100k	**99.53**	95.27	71.66
LSSL [21]	-	-	93.58	-
S4 [19]	-	-	93.96	**98.32**
FlexTCN [44]	373k	**99.53**	**97.67**	91.73
Ours	371k	**99.53**	97.45	**94.95**

Table 3. Time-series classification results.

of-the-art results on both sMNIST and pMNIST. For sCIFAR10 dataset, our model has outperformed all the comparative models except S4 [19]. Compared with the FlexTCN, which has a similar network base, our model improved the accuracy by 4%. These results show that our proposed model is suitable and effective for sequential images.

Time-series. We evaluated our model on time-series sequence datasets, character trajectories (CT) [1], and speech commands (SC) [59]. The results have been displayed in Tab. 3. In the relatively shorter MFCC features data, SMPConv achieved test accuracy similar to FlexTCN. To validate that our proposed model can model extremely long sequences, we conducted experiments on the SC-raw dataset, which has a sequence length of 16000. Similar to the sequence image classification result, our model outperformed FlexTCN with a large margin of +3%.

Compared to other models, our SMPConv has achieved considerably better performance for both sequential image and time-series classification. It ensures that our kernel representation is capable of handling long-term dependencies even in the case of a limited number of parameters.

Model	Params.	Accuracy
ResNet-44 [24]	660k	92.9
CKCNN-16 [44]	630k	72.1
FlexNet-16 [44]	670k	92.2
Ours	490k	**93.0**

Table 4. 2D image classification on CIFAR10.

4.3. Image classification

Image classification with the continuous kernel. We validated our SMPConv on a 2D image dataset, CIFAR10 [28], which is dominated by discrete convolutions, to show that the continuous kernel can capture spatial information as well. Similar to the experiments on sequential data, we followed the network design choice of FlexNet [44], where the kernel size is 33×33. More details are in Appx. 1.1.

As shown in Tab. 4, our continuous kernel representation model slightly outperforms ResNet, a discrete 3×3 convolution model, with a less number of parameters. It implies that our model is competitive and promising. Our model also showed better performance than MLP-based counterparts, CKCNN-16 and FlexNet-16, even when the parameters of ours were around 30% lesser. In addition, we have already identified the efficiency of our model in Tab. 1. These results suggest that our method is more suitable for kernel generation than MLP-based implicit formulations.

Large scale image classification. Finally, we tested our SMPConv on a large-scale ImageNet dataset [12], which contains more than one million training images and 50,000 validation images. For such a large dataset, the convolution kernels should be carefully trained to model complex data relationships accurately. Through such an experiment, therefore, we can validate that our `SMP` can represent a descriptive convolution kernel.

Firstly, we constructed large-scale variants of SMPConv architecture based on RepLKNet [13]. We replaced its discrete depth-wise separable convolution kernel with our `SMP`. In general, the larger the data and network, the larger the number of filters. To prevent excessive point position parameters depending on the number of filters, we shared the position of points over filters in large-scale settings. We empirically found that this position sharing has little effect on classification performance.

We proposed two variants of our model, SMPConv-T and SMPConv-B. Thanks to our efficient large kernel, we adjusted the number of channels and blocks so that our variants have a similar number of parameters to the previous models. The number of blocks and channels for each stage is [2, 2, 8, 2] and [96, 192, 384, 768] for SMPConv-T and [2, 2, 20, 2] and [128, 256, 512, 1024] for SMPConv-B, respectively. In RepLKNet-31B, the number of blocks and

Model	Params.	FLOPs	Top-1 Accuracy
ResNet-50 [24]	26M	4.1G	76.5
ResNext-50-32x4d [62]	25M	4.3G	77.6
ResMLP-S24 [54]	30M	6.0G	79.4
DeiT-S [55]	22M	4.6G	79.8
Swin-T [34]	28M	4.5G	81.3
TNT-S [22]	24M	5.2G	81.3
ConvNeXt-T [35]	29M	4.5G	82.1
SLaK-T [32]	30M	5.0G	**82.5**
SMPConv-T(ours)	27M	5.7G	**82.5**
DeiT-Base/16 [55]	87M	17.6G	81.8
Swin-B [34]	88M	15.4G	83.5
ConvNeXt-B [35]	89M	15.4G	83.8
SLaK-B [32]	95M	17.1G	**84.0**
RepLKNet-31B [13]	79M	15.3G	83.5
SMPConv-B(ours)	80M	16.6G	**83.8**

Table 5. 2D image classification on ImageNet-1K.

Models	radius	coordinate	Accuracy
A	-	-	90.92
B	✓	-	91.35
C	-	✓	92.47
SMPConv	✓	✓	**93.00**

Table 6. Ablation study on CIFAR10. A checkmark means that the component is a learnable. In case of SMPConv, for instance, both radius and coordinate are learnable parameters.

σ	0.05	0.2	0.3	0.5
Accuracy	**93.00**	92.24	91.84	91.51

Table 7. Classification results on CIFAR10 with different standard deviation σ of point location sampling distribution.

r	0.12	0.18	0.24	0.3
Accuracy	**93.00**	92.36	92.06	91.76

Table 8. Classification results on CIFAR10 with different initial radius r.

channels for each stage is [2, 2, 18, 2] and [128, 256, 512, 1024]. More experimental details are provided in Appx. 1.2.

As reported in Tab. 5, our models obtained competitive performance with fewer parameters than existing models. These results show that our kernel representation is promising for large-scale domains as well. Overall, our kernel representation is highly effective and descriptive.

4.4. Ablation

We performed various ablation studies with additional experiments on CIFAR10 image classification. First, we investigated the validity of learnable radius and coordinate. In Tab. 6, it showed that performance degradation occurs when either one or both components are set to non-learnable parameters. Remarkably, Model C, which set coordinate to trainable parameters, outperformed Model A by a considerable margin. Furthermore, Model B also had a slight performance gain. It suggests that even randomly distributed fixed weight points can increase their interpolation ability with trainable radius. These results indicate that training both coordinate and radius is valid.

N_p	4	8	16	32	64
Params.	250k	330k	490k	809k	1447k
Accuracy	92.56	92.28	**93.00**	92.84	92.21

Table 9. Classification results on CIFAR10 with different number of moving points N_p.

Next, we identified the effect of the initial position of the points by varying the σ, a standard deviation of point location sampling distribution. In Tab. 7, we observed that a small σ value, indicating initial positions of the points are gathered in the center of the kernel, leads to higher accuracy. This is because it is difficult to train large kernels from the beginning of training. Thus, large kernels can be effectively trained by starting with small kernels and expanding the receptive fields through moving points.

We also found that larger initial radius degrades the model performance as shown in Tab. 8. The large radius results in a large initial kernel size, which makes initial training difficult. Furthermore, it is also challenging to train a large area of the kernel dependent on a single weight point which is not optimized in the early stages of training. Both Tab. 7 and Tab. 8 empirically show that our initialization methods for `SMP` are effective.

As depicted in Tab. 9, we figured out that simply increasing the number of weight points N_p does not helpful for performance. It implies that a small number of points are enough to represent a proper convolution filter. Since the performance is influenced by the number of points, our method is also required tuning like common neural networks. However, the performance difference between CKCNN and FlexNet in Tab. 4 shows that MLP-based methods are severely influenced by architectural settings. That is, they have extensive search space, such as depth, width, and activation function, so they typically require more tuning than our method. Moreover, the impact of the number of points is not particularly significant in that even the worst ($N_p = 64$) slightly outperforms the FlexNet (acc=92.20).

4.5. Visualization

Finally, we analyze our `SMP` by visualizing filters trained on CIFAR10. In the first column of Fig. 4, we can observe the trained weight points' position. In our method, point lo-

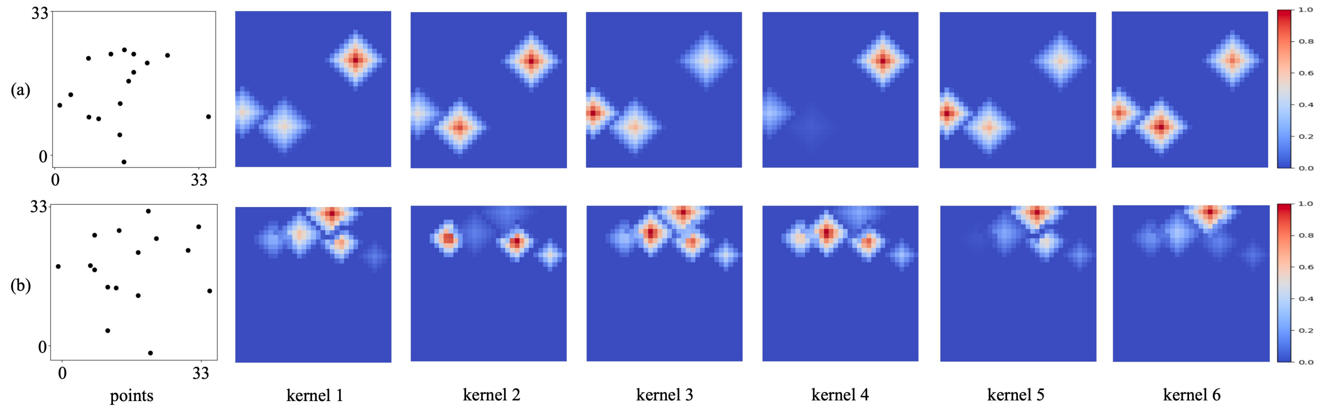

Figure 4. Visualization of kernels. Each row shows the location of points and first 6 kernels of a filter. For ease of visualization, the kernels are first subjected to the absolute value operation and then normalized to a range of [0,1].

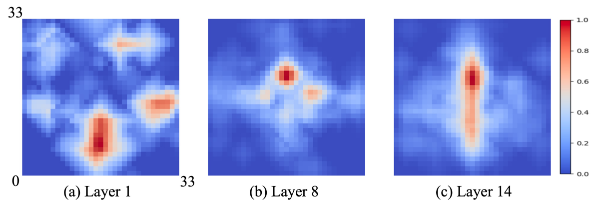

Figure 5. Normalized sum of the absolute value of trained filters. (a), (b), and (c) are top, middle, and bottom layers, respectively.

cations p_i are mainly sampled near the center of the kernel for stable training. It shows that the points spread out for optimal kernel representation over the training process, as we argued, and thus the receptive fields are not limited to a small part. Also, we can figure out that there are square patterns caused by Eq. (2) in the kernels, where each square has its own area. This suggests that although the radius parameters are initialized with small values, the values are individually increased and optimized for each corresponding weight point during training.

Observing visualized convolution kernel in Fig. 4, kernels from the same filter share the receptive fields. It allows a single filter to focus on the shared area. Furthermore, as illustrated in Fig. 5, SMPConv has large adaptive receptive fields which are not conventional square or rectangular shapes. This is because it consists of optimized filters with their own small and large receptive fields. Thus, our method can handle not only global information but also local details.

5. Conclusion and discussion

In this paper, we present a method to build a continuous convolution. We propose using point representations, where each point has the weight parameters, coordinates, and radius to learn. By connecting the points, we can implement a continuous function, which can be utilized to construct convolutional kernels. We have provided extensive experimental results, showing that drop-in replacement in the existing training pipeline without bells and whistles improved the performance by a safe margin. We also show that a continuous convolution can be effectively utilized in a large-scale experiment. We expect more research and development in this direction.

Although promising, there are many rooms to be improved. Due to the limited computational budget, we could not conduct sufficient experiments in the large-scale experiment. The experimental results provided in this manuscript resulted from a few runs. As trial and error are essential in the machine learning development process, we plan to find optimal configurations and training techniques to enhance the performance of the proposed method.

We also observed that the learned kernels often show sparse patterns depending on the tasks. It is well aligned with the success of dilated convolution or its variants, and our methods automatically learn proper sparsity during the training process on specific tasks. Adding prior knowledge through training and regularization techniques would further improve performance, especially for tasks requiring longer-term dependency modeling.

Acknowledgments

We thank Usman Ali for valuable discussions. This research was supported by the Ministry of Science and ICT (MSIT) of Korea, under the National Research Foundation (NRF) grant (2022R1F1A1064184, 2022R1A4A3033571), Institute of Information and Communication Technology Planning Evaluation (IITP) grants for the AI Graduate School program (IITP-2019-0-00421), and the BK21 FOUR Project.

References

[1] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018. 1, 6

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. 6

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*, 2018. 6

[4] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated recurrent neural networks. In *NeurIPS*, 2017. 5, 6

[5] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018. 6

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 2

[7] 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. *TPAMI*, 40(4):834–848, 2017. 3

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 3

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017. 3, 5

[10] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, pages 472–488. Springer, 2016. 1

[11] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *NeurIPS*, 2019. 6

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 3, 6

[13] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *CVPR*, pages 11963–11975, 2022. 1, 3, 4, 6, 7

[14] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *CVPR*, pages 13733–13742, 2021. 4

[15] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *CVPR*, pages 12124–12134, 2022. 3

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3

[17] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021. 2

[18] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In *NeurIPS*, pages 1474–1487, 2020. 6

[19] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 6

[20] Albert Gu, Caglar Gulcehre, Thomas Paine, Matt Hoffman, and Razvan Pascanu. Improving the gating mechanism of recurrent neural networks. In *ICML*, pages 3800–3809. PMLR, 2020. 6

[21] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *NeurIPS*, pages 572–585, 2021. 6

[22] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *NeurIPS*, pages 15908–15919, 2021. 7

[23] Ismail Khalfaoui Hassani, Thomas Pellegrini, and Timothée Masquelier. Dilated convolution with learnable spacings. *arXiv preprint arXiv:2112.03740*, 2021. 3

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3, 6, 7

[25] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 1

[26] 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. 3

[27] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *NeurIPS*, pages 6696–6707, 2020. 6

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[29] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015. 5

[30] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *CVPR*, pages 5457–5466, 2018. 6

[31] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 3

[32] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022. 1, 2, 3, 7

[33] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, pages 8895–8904, 2019. 3

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021. 3, 7

[35] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, pages 11976–11986, 2022. 3, 7

[36] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *ICCV*, 2019. 1

[37] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pages 4460–4470, June 2019. 2

[38] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2

[39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 2

[40] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, June 2019. 2

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 3

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 3

[43] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*, pages 5301–5310. PMLR, 2019. 2

[44] David W Romero, Robert-Jan Bruintjes, Jakub M Tomczak, Erik J Bekkers, Mark Hoogendoorn, and Jan C van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *arXiv preprint arXiv:2110.08059*, 2021. 1, 2, 3, 5, 6

[45] David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021. 1, 2, 3, 6

[46] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *NeurIPS*, 2019. 6

[47] T Konstantin Rusch and Siddhartha Mishra. Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. *arXiv preprint arXiv:2010.00951*, 2020. 6

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[49] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, pages 7462–7473, 2020. 1, 2

[50] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, pages 5459–5469, 2022. 2

[51] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114. PMLR, 2019. 3

[52] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, pages 6411–6420, 2019. 1, 3

[53] Nergis Tomen and Jan C van Gemert. Spectral leakage and rethinking the kernel size in cnns. In *ICCV*, pages 5138–5147, 2021. 3

[54] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *TPAMI*, 2022. 7

[55] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357. PMLR, 2021. 3, 7

[56] Trieu Trinh, Andrew Dai, Thang Luong, and Quoc Le. Learning longer-term dependencies in rnns with auxiliary losses. In *ICML*, pages 4965–4974. PMLR, 2018. 6

[57] Cristina Vasconcelos, Kevin Swersky, Mark Matthews, Milad Hashemi, Cengiz Oztireli, and Andrea Tagliasacchi. Cuf: Continuous upsampling filters. *arXiv preprint arXiv:2210.06965*, 2022. 1

[58] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, pages 2589–2597, 2018. 1, 2, 3

[59] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018. 1, 6

[60] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, pages 9621–9630, 2019. 3

[61] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. 1

[62] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 7

[63] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, pages 5438–5448, 2022. 2, 3

[64] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, pages 87–102, 2018. 1

[65] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM ToG*, 35(6):1–12, 2016. 1

[66] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021. 2

[67] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. *arXiv preprint arXiv:2205.14330*, 2022. 2, 3