Point-to-Spike Residual Learning for Energy-Efficient 3D Point Cloud Classification

Qiaoyun Wu[1,2,3], **Quanxiao Zhang**[1,2,3], **Chunyu Tan**[1,2,3], **Yun Zhou**[1,4], **Changyin Sun**[1,2,3]
[1]School of Artificial Intelligence, Anhui University
[2]Engineering Research Center of Autonomous Unmanned System Technology, Ministry of Education
[3]Anhui Provincial Engineering Research Center for Unmanned System and Intelligent Technology
[4]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
wuqiaoyun@ahu.edu.cn, zhouy@ahu.edu.cn

Abstract

Spiking neural networks (SNNs) have revolutionized neural learning and are making remarkable strides in image analysis and robot control tasks with ultra-low power consumption advantages. Inspired by this success, we investigate the application of spiking neural networks to 3D point cloud processing. We present a point-to-spike residual learning network for point cloud classification, which operates on points with binary spikes rather than floating-point numbers. Specifically, we first design a spatial-aware kernel point spiking neuron to relate spiking generation to point position in 3D space. On this basis, we then design a 3D spiking residual block for effective feature learning based on spike sequences. By stacking the 3D spiking residual blocks, we build the point-to-spike residual classification network, which achieves low computation cost and low accuracy loss on two benchmark datasets, ModelNet40 and ScanObjectNN. Moreover, the classifier strikes a good balance between classification accuracy and biological characteristics, allowing us to explore the deployment of 3D processing to neuromorphic chips for developing energy-efficient 3D robotic perception systems.

Introduction

3D point clouds have become a popular representation form of physical entities in the last few years, and have a wide range of applications in computer vision and robotics. Among various 3D point cloud processing researches, the classification of 3D point clouds is of great importance. Over the past decade, 3D point cloud classification methods based on deep learning have demonstrated high and relatively stable classification accuracy on benchmark datasets (Thomas et al. 2019; Choy, Gwak, and Savarese 2019). However, these methods require a large number of multiply-accumulate (MAC) operations, resulting in high computational cost and high energy consumption, which also limits the usage of these methods in embedded systems of robots.

Spiking neural networks, as the third generation of neural networks, simulate the information encoding and transmitting of biological neurons, which have achieved extremely high computing efficiency on neuromorphic hardwares, e.g., Loihi (Davies et al. 2018). Therefore, it would be natural to consider designing SNNs for 3D point cloud processing. SNNs are composed of spiking neurons and each neuron provides spike signals only when its membrane potential exceeds a certain threshold. The generated spike signals would be propagated to the neurons in the next layer, contributing to the membrane potential charging. Based on this, the communication between neurons occurs through sparse spike signals, and the process of input spikes in neurons only involves simple accumulate (AC) operations on the membrane potential. Compared with traditional artificial neural networks (ANNs), which require abundant MAC operations among real-value signals, SNNs show excellent energy consumption advantages (Kim et al. 2020).

To date, SNNs have been successfully applied to a variety of areas in the field of neuromorphic computing, such as visual image classification and tracking, robot decision control (Oikonomou, Kansizoglou, and Gasteratos 2023), etc. In these applications, SNNs achieve performance comparable to conventional ANNs with low power consumption and high biological rationality. Following these successes, it is desirable to extend SNNs to 3D point cloud processing for advanced 3D vision tasks with embedded and energy-saving requirements. To achieve the extension, two key issues deserve much attention: (1) The generation of spike signals should be closely related to the geometry of 3D point clouds. (2) Spike sequences in high-dimensional hidden space should have the ability to represent different geometric contexts of 3D points.

In this paper, we focus on developing SNNs for energy-efficient 3D point cloud classification. To address the first issue, we exploit the spatial localization property that kernel point convolutions (Thomas et al. 2019) maintain and the spiking processing mechanism of Integrate-Fire (IF) neurons (Bulsara et al. 1996). We define spatial-aware kernel point spiking neurons to relate spiking generation to point position in 3D space. The neurons take spike sequences from local structures as inputs and replace abundant MAC operations in kernel point convolutions (KPConv) with simple kernel weight AC operations for membrane potential charging. Driven by the sparse nature of spiking inputs and the AC replacement, the proposed neurons present impressive power efficiency compared to conventional KPConv (Thomas et al. 2019) and are promising for the implementation in neuromorphic architectures. For the second issue, we design 3D spiking residual blocks by integrating our kernel

point spiking neurons into residual blocks. The proposed residual blocks contrive different cross-layer connections for improving information flow efficiency and feature representation. Eventually, we stack the 3D spiking residual blocks to further improve the nonlinear representation ability, and combine fully connected layers to build a point-to-spike residual learning network for 3D classification called P2SResLNet. The main contributions are as follows:

- We define spatial-aware kernel point spiking neurons for 3D local structure perception, which realizes the combination of spiking neurons and conventional point convolutions for the first time. The neurons operate on sparse spike signals and replace a large number of MAC with AC, which significantly reduces the energy consumption of convolution computation for 3D point clouds.
- We design 3D spiking residual blocks with different cross-layer connections for improving the information flow efficiency and the information expression capability of spike sequences in high-dimensional hidden spaces.
- We build a deep point-to-spike residual learning network for 3D point cloud classification, which can significantly reduce energy consumption while maintaining high accuracy on two benchmark datasets, ModelNet40 (Wu et al. 2015) and ScanObjectNN (Uy et al. 2019).

Related Work

Feature Learning on Point Clouds

Existing researches on deep learning methods for point clouds fall into two broad categories. One is about learning on the basis of structured grids, which requires converting unstructured point clouds into a structured form. These methods can be further divided into two types, i.e., view-based methods (Tatarchenko et al. 2018) and voxel-based methods (Choy, Gwak, and Savarese 2019). View-based methods project 3D points into 2D images from different views, and then some well-designed 2D convolutional neural networks can be exploited to extract features from these images. However, the projection brings extensive occlusions, which may result in severe performance degradations. Voxel-based methods project 3D points into 3D voxels, and then design 3D convolutional neural networks to process these voxels. These methods have shown great point cloud processing performances. However, the voxel resolution limitation of some computing configurations restricts the processing precision and scale of point clouds.

The other category is about learning on the basis of unstructured inputs, including graph-based learning and direct point-based learning. Graph-based learning generally constructs graphs based on input point clouds followed by proposing graph neural networks for feature extraction (Zhang et al. 2023). These methods learn based on edge relationships, which can easily ignore the local deformation in Euclidean space. In addition, constructing a graph with all the points as graph nodes requires considerable computations. In this paper, we focus on the point-based learning, which designs various multi-layer perceptrons (MLPs) or convolutions for point feature learning. The pioneering works are PointNet (Qi et al. 2017a) and PointNet++ (Qi et al. 2017b). PointNet learns point-by-point features through MLPs and extracts global features through a maximum pooling, where abundant local structure information is ignored. On this basis, PointNet++ is developed to hierarchically extract features at different scales. PointMLP (Ma et al. 2022) and PointStack (Wijaya, Paek, and Kong 2022) are latest point cloud processing methods, which follow the design philosophy of PointNet++. PointMLP uses a feedforward residual MLP network to learn point cloud representation, which transmits information and extracts features through multiple fully connected layers. PointStack designs multi-resolution learning and learnable pooling for extracting high-semantic and high-resolution point features. On the other hand, PointCNN (Li et al. 2018) propose a χ-Conv operator to weight and permute input points, and then apply typical convolutions to the χ-transformed features. KPConv (Thomas et al. 2019) designs kernel point convolutions with variable number and position of kernel points, which presents more flexibility than standard convolutions and inspires our work.

Spiking Neural Networks

Compared to conventional artificial neural networks, spiking neural networks possess great potentials in mimicking biological neuron dynamics to achieve high performances with low power consumptions. Spiking neurons are the key components of SNNs and the commonly used spiking neurons in SNNs (Kim et al. 2020; Fang et al. 2021) are Integrate-Fire (IF) (Bulsara et al. 1996) and Leaky-Integrate-Fire (LIF) (Gerstner and Kistler 2002). Both types of neurons accumulate membrane potentials by receiving synthesized currents and generate spikes when the membrane potential exceeds a threshold. IF neurons can be regarded as ideal integrators, maintaining a constant voltage in the absence of spike inputs, while LIF neurons gradually decay in voltage in the absence of inputs. Compared to LIF, IF requires less memory access and energy consumption. Some other spiking neurons, such as the second-order dynamic neurons (Zhang et al. 2022) and the multi-compartment spiking neurons (Sun et al. 2023), can better represent dynamic characteristics of biological neurons than IF but with several predefined hyper-parameters.

There are two main schemes to obtain deep spiking neural networks: one based on an ANN-to-SNN conversion, and the other based on backpropagation. The ANN-to-SNN conversion scheme firstly trains a ANN-based baseline, and then converts the baseline to a SNN-based model by directly substitute ReLU activations for spiking neurons with some normalization and threshold strategies (Cao, Chen, and Khosla 2015). However, models based on the conversion usually require a large number of time steps to approximate the performance of original ANN-based baselines, inevitably resulting in massive delays. Essentially, the ANN-to-SNN conversion scheme avoids the difficulty of training SNNs directly, which results from the non-differentiable of spiking neurons in SNNs. In the second scheme, researchers directly train SNNs from scratch by designing surrogate gradients for backpropagation (Lee et al. 2020), or using the gradients

with respect to the membrane potentials (Zhou et al. 2021). Models based on direct training present a clear advantage in reducing spiking time latency and are more suitable to practical applications. Considering the above, we use the direct training scheme to get our point-to-spike residual learning network in this paper.

Methods

We propose a deep point-to-spike residual learning network for 3D point cloud classification. We first combine spiking neurons with conventional kernel point convolution operations and define spatial-aware kernel point spiking neurons for capturing local geometric structures. Then, we integrate the spiking neurons into conventional residual blocks to design 3D spiking residual blocks, improving 3D point feature learning based on spike sequences. Finally, we stack several 3D spiking residual blocks and fully connected layers to build our classifier.

Kernel Point Spiking Neurons

Inspired by the combination of spiking neurons and convolutions in 2D image processing (Zhou et al. 2023), we first define spatial-aware kernel point spiking neurons to combine spiking neurons (Bulsara et al. 1996) with kernel point convolutions (Thomas et al. 2019) in point cloud processing.

Integrate-fire spiking neurons. We choose the Integrate Fire (IF) spiking neuron (Bulsara et al. 1996) in this paper, which is the simplest neuron type. The dynamic model of a IF neuron can be described as:

$$u_t = V_{t-1} + In_t \tag{1}$$

$$S_t = \Theta(u_t - V_{th}) \tag{2}$$

$$V_t = u_t(1 - S_t) + V_{reset}S_t \tag{3}$$

Where u_t represents the membrane potential of the neuron that has undergone neuron dynamics at time step t. $t \in [0, T)$, where T, the time latency, is an important hyper-parameter defined in IF neurons. V_{t-1} represents the membrane potential of the neuron that has undergone a spiking triggering judgment at time step $t-1$, and In_t represents the input current at time step t. In Equation 2, $\Theta(\cdot)$ is the spiking triggering judgment at time step t . When the membrane potential u_t exceeds the firing threshold V_{th}, the neuron will generate a spike. Therefore, $\Theta(v)$ is designed as the Heaviside step function. When $v \geq 0$, its value is 1; otherwise, it is 0. In Equation 3, V_t is the membrane potential of the neuron that has undergone the spiking triggering judgment at time step t. If no spike is generated, $V_t \equiv u_t$; otherwise, it is set to the reset potential V_{reset}.

Kernel point convolutions. The design of 3D kernel point convolutions (Thomas et al. 2019) is inspired by image convolutions, which captures local geometric structures by building the the relations between kernel points and neighborhood points. The 3D kernel point convolution is formulated as:

$$(F * g)(x) = \sum_{x_i \in R_x} g(x_i - x) f_i \tag{4}$$

Where x_i and f_i denote the 3D position of point i and its corresponding feature, respectively. $R_x = \{x_i \in P | \|x_i - x\| \leq r\}$ is the set of all points in a sphere, which takes x as the center and r as the radius. $P \in R^{N*3}$ is the input point set and $F \in R^{N*D}$ is the corresponding feature set. The kernel function g takes the neighborhood centered at x as input. We define $y_i = x_i - x$, and the domain of the kernel function g is the sphere $S_r = \{y \in R^3 | \|y\| \leq r\}$. KPConv selects K points $\{\tilde{x}_k\}_{k=1}^K \subset S_r$ as kernel points, and $\{W_k\}_{k=1}^K \subset R^{D_{in}*D_{out}}$ as the associated weight matrix set. For each y_i, the kernel function is defined as:

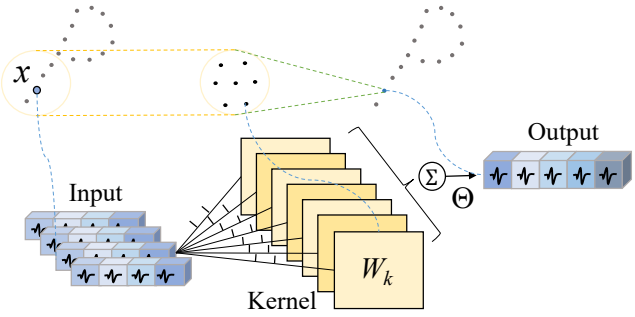

Figure 1: The dynamics of a spatial-aware kernel point spiking neuron, denoted as KPS neuron.

$$g(y_i) = \sum_{k=1}^{K} c(y_i, \tilde{x}_k) W_k \tag{5}$$

Where $c(\cdot)$ is the correlation function, representing the connection between $\tilde{x}_k$ and y_i. The closer the two values are, the higher the $c(\cdot)$ is. $c(\cdot)$ is defined as:

$$c(y_i, \tilde{x}_k) = max(0, 1 - \frac{\|y_i - \tilde{x}_k\|}{\sigma}) \tag{6}$$

Where σ is a hyper-parameter and KPConv sets it according to the input density.

Kernel point spiking neurons. We combine the IF neuron with the kernel point convolution to define a spatial-aware kernel point spiking (KPS) neuron as:

$$\begin{aligned} u_t^l(x) &= V_{t-1}^l(x) + In_t^l(x), \\ In_t^l(x) &= \sum_{x_i \in R_x} g^l(x_i - x)\Theta(u_t^{(l-1)}(x_i) - V_{th}), \\ S_t^l(x) &= \Theta(u_t^l(x) - V_{th}), \\ V_t^l(x) &= u_t^l(x)(1 - S_t^l(x)) + V_{reset}S_t^l(x), \end{aligned} \tag{7}$$

Where $u_t^l(x)$ denotes the membrane potential of the neuron that has undergone neuron dynamics at the 3D position x of the l-th layer at time step t. V_{t-1}^l represents the membrane potential of the neuron that has undergone a spiking triggering judgment at the position x of the l-th layer at time step $(t-1)$. $In_t^l(x)$ indicates the input current at the position x of the l-th layer at time t. The kernel function $g^l(\cdot)$ is defined by the Equation 5 and when the number of kernel points K is between 17 and 23, there is little performance difference. We empirically set K to 20. $u_t^{l-1}(x)$ represents the membrane potential of the neuron that has undergone neuron dynamics at the position x of the $(l-1)$-th layer at

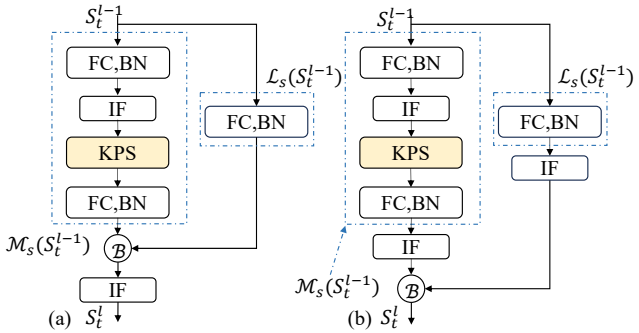

Figure 2: 3D spiking residual blocks.

time step t. $\Theta(\cdot)$ is the Heaviside step function. Hence, the computation of $In_t^l(x)$ can be converted to simple AC operations and the weight accumulation occurs only when the neighbor of x generate a spike in the $(l-1)$-th layer. Namely, the sparse AC operations here replaces a large number of MAC operations in Equation 4, which significantly reduces the computation cost. Figure 1 shows the dynamics of the proposed kernel point spiking neuron. At the position x of the l-th layer during the time latency T, after receiving the spike input in the neighborhood R_x of the $(l-1)$-th layer, denoted as $\{S^{l-1}(x_i)\}_{i=1}^{|R_x|}$, the neuron outputs a spike sequence $S^l(x)$ after the spiking triggering judgement $\Theta(\cdot)$.

3D Spiking Residual Block

Spiking neural networks are based on binary spikes rather than continuous floats, which causes great difficulties in training deep SNNs. On the other hand, residual learning in ANNs has been proved to be an effective technique, which can alleviate the problems of gradient disappearance and model degradation caused by network deepening. We therefore design a 3D spiking residual block based on the 3D kernel point spiking neuron and the residual operation (He et al. 2016). The basic l-th residual block is formulated as:

$$h^l = \mathcal{B}(\mathcal{M}(h^{l-1}), \mathcal{L}(h^{l-1})) \tag{8}$$

Where h^{l-1} and h^l are both feature representations in high-dimensional hidden space, and denote the input and output of the l-th residual block, respectively. $\mathcal{M}(\cdot)$ and $\mathcal{L}(\cdot)$ are both feature mapping structures, which are mainly composed of convolution layers, batch normalization layers, activation function layers. $\mathcal{B}(\cdot)$ is the connection function, which is the element-by-element addition operation in (He et al. 2016). Inspired by the basic residual block, we design a 3D spiking residual block in two forms as:

$$S_t^l = IF(\mathcal{B}_s(\mathcal{M}_s(S_t^{l-1}), \mathcal{L}_s(S_t^{l-1}))) \tag{9}$$

$$S_t^l = \mathcal{B}_s(IF(\mathcal{M}_s(S_t^{l-1})), IF(\mathcal{L}_s(S_t^{l-1}))) \tag{10}$$

Where S_t^{l-1} and S_t^l are both spike sequences in high-dimensional hidden space at time t, and are the input and output of the l-th 3D spiking residual block, respectively. The structures of our 3D spiking residual blocks are shown in Figure 2. The main difference is the combination of IF neurons and feature mapping structures. We select the optimal structure design in Figure 2 (b) based on the experiments on the benchmark dataset in Sec. . Moreover, considering the binary property of input spike sequences, we design three connection functions for $\mathcal{B}_s$ as shown in Table 1 and evaluate their performances to choose the most appropriate one (ADD) in Sec. .

$\mathcal{B}_s$	Implementation
ADD	$IF(\mathcal{M}_s(S_t^{l-1})) + IF(\mathcal{L}_s(S_t^{l-1}))$
AND	$IF(\mathcal{M}_s(S_t^{l-1})) \wedge IF(\mathcal{L}_s(S_t^{l-1}))$
IAND	$\neg IF(\mathcal{M}_s(S_t^{l-1})) \wedge IF(\mathcal{L}_s(S_t^{l-1}))$

Table 1: Three connection functions for $\mathcal{B}_s$.

Point-to-Spike Residual Learning Network

We build a point-to-spike residual learning network (P2SResLNet) for 3D point cloud classification based on stacking 13 proposed spiking residual blocks and fully connected layers in Figure 3. The input is a 3D point set $P' \in R^{T*N*4}$, where T represents the spiking time latency, N denotes the number of points, and 4 represents the feature dimension, including the 3D coordinates of points and the constant 1 by following (Thomas et al. 2019). To convert these real-valued features into spike-form, the first part of P2SResLNet takes advantage of conventional KPConv in a basic block to extract local features for each spatial point, followed by using IF neurons to process each element of the features to generate spike sequences during T time steps. Taking the generated spike sequences as inputs, the second part of P2SResLNet is stacked with four similar structures, each consisting of a point cloud down-sampling operation as in (Thomas et al. 2019), and three stacked 3D spiking residual blocks. The third part, a classification head, is designed to predict the category Y, which can be realized in four forms:

$$Y = AvgPooling(FC(O^L)) \tag{11}$$

$$Y = AvgPooling(FC(IF(O^L))) \tag{12}$$

$$Y = FC(AvgPooling(O^L)) \tag{13}$$

$$Y = FC(AvgPooling(IF(O^L))) \tag{14}$$

Where O^L is the output of the L-th 3D spiking residual block (i.e., the output of the second part of P2SResLNet), which can be decoded in two forms: one based on the spike sequences of the last spiking neurons in the L-th 3D spiking residual block, and the other based on the accumulated membrane potentials. AvgPooling after FC, in Equation 11 and 12, can be considered as classifying based on averaging neuron dynamics, resulting in a large number of parameters of FC. AvgPooling before FC, in Equation 13 and 14, can reduce the parameters of FC. We mainly adopt the accumulated membrane potentials for O^L and the head in Equation 13 based on the experimental analysis in Sec. .

We update the parameters of P2SResLNet via cross entropy loss and gradient back-propagation. Note that the derivative of the Heaviside step function in Equation 7 is equal to the Dirac delta function, and its direct use for gradient descent will make the training process extremely unstable. Following (Fang et al. 2020), we use the surrogate function, $\Theta'(v) \triangleq \sigma'(v)$, in the gradient back-propagation process, where $\sigma(v)$ is a smooth and continuous function with a shape similar to $\Theta(v)$. The arctan is used here.

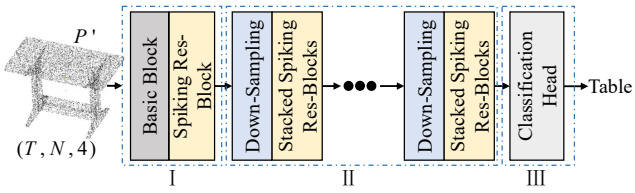

Figure 3: The architecture of P2SResLNet, which consists of three parts. P2SResLNet takes as input a 3D point set and finally predicts a category of the point set.

Experiments and Results

Experimental Settings

We evaluate the performance of 3D point cloud classification on both the synthetic dataset ModelNet40 (Wu et al. 2015) and the real dataset ScanObjectNN (Uy et al. 2019). ModelNet40 contains 40 different object categories, each of which contains a large number of 3D object instances. The training set contains $9,843$ instances, and the testing set contains $2,468$ instances. ScanObjectNN is constructed based on real-world scanning, which is characterised by varying degrees of data missing and noise contamination. The entire dataset contains 3D objects from 15 categories with $11,416$ samples as training set and $2,882$ samples as testing set. In terms of the evaluation metrics, we use the point cloud classification overall accuracy (OA) and the category mean accuracy (mAcc) by following (Wijaya, Paek, and Kong 2022).

The proposed network is designed based on the the biological plausibility and it is desirable to deploy it directly on neuromorphic hardwares. However, designing RC circuits on a neuromorphic hardware for 3D point kernel spiking neurons requires extra engineering efforts, which is out of the scope of this work. On the other hand, to maintain high classification accuracy, there are still some MAC operations in the network, such as the first part of the network and the kernel computation in Equation 5, which are difficult to implement on neuromorphic hardwares. Hence, our experiments are conducted on a PC with the 11-th Gen Intel $i7-11700K$ $3.60GHz$ 16-core processor and a NVDIA GeForce RTX 3070 GPU. Our implementation is based on PyTorch and SpikingJelly (Fang et al. 2020). We update the network parameters using the SGD optimizer and the learning rate is initialized to 10^{-3}. The sampling radius of the first point cloud down-sampling layer is an important hyper-parameter. In follow-up experiments, we set it for ModelNet40 to 0.15, for ScanObjectNN to 0.3 by default. For the spiking neurons, we set the time latency T to 1 by default.

Ablation Study

We first conduct ablation experiments on ModelNet40 to determine the final architecture of P2SResLNet.

Ablation on 3D spiking residual block designs. Figure 2 presents two types of spiking residual blocks with the ADD connection function by default. We build two classification networks based on the two blocks, denoted as P2SResLNet-A and P2SResLNet-B, respectively. Table 2 reports the evaluation results on ModelNet40 testing set. P2SResLNet-B presents a 1.2% improvement in the OA and a 1.8% improvement in the mAcc compared to P2SResLNet-A (90.6% $vs.$ 89.4% and 89.2% $vs.$ 87.4%). Hence, we choose the block structure in Figure 2 (b) for building our final classifier in follow-up experiments.

Model	OA(%)	mAcc(%)
P2SResLNet-A	89.4	87.4
P2SResLNet-B	**90.6**	**89.2**

Table 2: Ablation study on the spiking residual block designs on the ModelNet40 dataset.

Connection	OA(%)	mAcc(%)
ADD	**90.6**	**89.2**
AND	89.2	88.2
IAND	89.0	87.5

Table 3: Ablation study on the connection functions on the ModelNet40 dataset. P2SResLNet with the ADD connection function presents the highest classification accuracy.

Ablation on connection functions. Table 1 presents three types of connection functions for $\mathcal{B}_s(\cdot)$, including ADD, AND, and IAND. We build networks based on these functions. As shown in Table 3, P2SResLNet with the ADD, achieves the highest overall classification accuracy than others. We also visualize the average firing rate of each block in these networks. The firing rate of a block is defined as $\frac{n_s}{(T*SN)}$, where n_s is the total number of spikes of the block during the default time latency T and SN represents the number of spiking neurons in the block. We use the SpikingJelly to obtain the average spike firing rate of each block. We can see in Figure 4 that P2SResLNet with the IAND provides much higher firing rates in majority of the blocks. However, its classification performance in Table 3 is the worst. The firing rate curve of P2SResLNet with the AND is mostly below the other two curves, with a large variation range. P2SResLNet-ADD has a moderation performance in terms of both the average firing rate and the variation range.

Ablation on classification heads. The third part of our classification network, i.e. the classification head, plays a

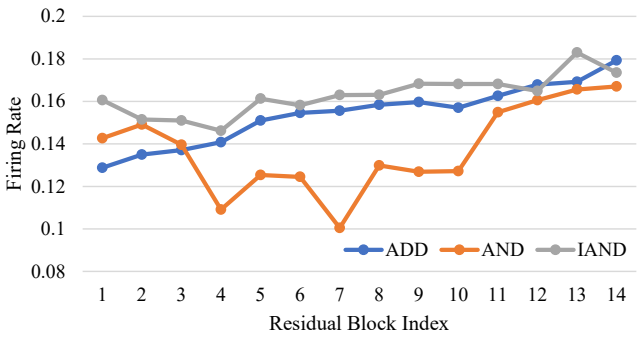

Figure 4: Firing rate distribution with respect to residual blocks for different connection functions on ModelNet40.

Classification Head	OA(%)	mAcc(%)
FC-Avgpooling	90.0	88.5
IF-FC-Avgpooling	89.6	87.7
Avgpooling-FC	**90.6**	**89.2**
IF-Avgpooling-FC	89.6	87.8

Table 4: Ablation study on the classification heads on the ModelNet40 dataset. P2SResLNet with the Avgpooling-FC head presents the highest classification accuracies.

Decoding Scheme	OA(%)	mAcc(%)
Spiking Sequence	90.4	88.5
Membrane Potential	**90.6**	**89.2**

Table 5: Ablation study on the decoding schemes on the ModelNet40 dataset. P2SResLNet using the membrane potential decoding presents better classification accuracies.

significant role in the overall performance, which has four forms as demonstrated in Equation 11-14. We provide the ablation evaluation on classification heads in Table 4. Both IF-Avgpooling-FC and IF-FC-Avgpooling use IF before other operations, and hence their FC are designed for spike sequences. Both Avgpooling-FC and FC-Avgpooling directly process the accumulated membrane potentials of the last spiking neurons in the L-th 3D spiking residual block. The results show that Avgpooling-FC with minimal parameters outperforms other heads slightly.

Ablation on decoding schemes. We provide two decoding schemes for O_L in Sec. . The evaluation results on the decoding schemes with the Avgpooling-FC classification head by default are presented in Table 5. We observe that the accumulated membrane potential for decoding outperforms the spiking sequence output scheme. It is considered that the membrane potentials, as floating-point numbers, contain more useful information for classification, resulting in more precise interpreting.

Ablation on time latency. The time latency T is an important hyper-parameter in SNNs and our evaluation results on the time latency are presented in Table 6. We observe that our network has the highest accuracy at one time step, and the classification performance difference is slight, when T is between 4 and 8. The results are inconsistent with the conclusion in SNN-based image classification (Fang et al. 2021), where longer time latency brings better classification accuracy. We consider there are two reasons for this. One is related to the dataset. 3D classification benchmark datasets usually lack temporal information compared to the DVS-based datasets in (Fang et al. 2021). Therefore, a larger T in our work may mean more redundancy rather than more useful information. The other is related to the latency setting. The latency setting of SNN-based image classification is invariably within a wide range, e.g., between 1 and 2000. However, it is difficult to set T greater than 10 in 3D training due to the system configuration limitation. Hence, the possibility of better performance with a large T cannot be ruled out in our network. In fact, a smaller T is more in line with some real application requirements.

Latency	OA(%)	mAcc(%)
1	**90.6**	**89.2**
4	88.7	86.6
6	88.6	86.5
8	88.9	86.5

Table 6: Ablation study on the time latency on the ModelNet40 dataset. P2SResLNet with $T = 1$ presents the highest classification accuracies.

Comparison with SOTAs

Table 7 presents the comparison results from three types of classification networks on two benchmark datasets, including ANN-based networks, a ANN-to-SNN network, and a SNN-based network. PointNet (Qi et al. 2017a), PointNet++ (Qi et al. 2017b), PointCNN (Li et al. 2018), KPConv (Thomas et al. 2019), PointStack (Wijaya, Paek, and Kong 2022), and PointMLP (Ma et al. 2022) are representative ANN-based networks here. For the ANN-to-SNN network, we follow (Cao, Chen, and Khosla 2015) to convert an ANN-based method into SNN-form. We choose KPConv, since our work is directly inspired by it. For the conversion, we first set the bias used in KPConv to 0 and then replace the activation function, LeakyReLU, with IF neurons. In addition, we use average pooling instead of max pooling to get the final SNN architecture, denoted as KPConv-SNN. Note that we do not train KPConv-SNN from scratch. Instead, we directly apply the pretrained parameters of KPConv (Thomas et al. 2019) to KPConv-SNN with the time latency $T = 40$, which is set based on a large number of experiments. Different from the above, SNN-based networks require training from scratch. The representative SNN-based network is our work with the time latency $T = 1$.

It is not surprising that ANN-based networks, e.g., KPConv, PointStack, and PointMLP, consistently outperform our SNN-based network, since floating-point numbers in ANNs can transfer more meaningful information than binary spikes in SNNs. The goal of this work is exploring effective ways to reduce the energy consumption during 3D point cloud processing while maintaining high classification accuracy. The proposed network achieves the overall accuracies of 90.6% and 81.2% on two benchmark datasets, respectively, which are very close to the performances of KPConv (Thomas et al. 2019) (our ANN-based baseline). We also provide their energy consumption comparison theoretically in Sec. , which demonstrates our network has the advantage of ultra-low energy consumption. In addition, the comparison between KPConv-SNN and ours indicates that the proposed point-to-spike residual learning network can provide a proper balance between classification accuracy and spike-based biological characteristics.

Theoretical Energy Consumption Calculation

In this section, we investigate energy efficiency of our SNN-based network compared to the ANN-based baseline (Thomas et al. 2019). We consider computing ener-

Model	Type	ModelNet40 OA/mAcc(%)	ScanObjectNN OA/mAcc(%)
PointNet	ANN	89.2/86.0	68.2/63.4
PointNet++	ANN	92.0/89.1	77.9/75.4
PointCNN	ANN	92.5/88.1	78.1/75.1
KPConv	ANN	92.1/90.7	85.3/83.7
PointStack	ANN	93.3/89.6	**86.9/85.8**
PointMLP	ANN	**94.1/91.5**	85.4/83.9
KPConv-SNN	ANN-to-SNN	70.5/67.6	43.9/38.7
P2SResLNet	SNN	90.6/89.2	81.2/79.4

Table 7: Comparison of various models on ModelNet40 and ScanObjectNN.

gy consumptions of the proposed kernel point spiking layer of our network and the conventional kernel point convolution layer of the baseline, where accumulate (AC) operations and multiply-accumulate (MAC) operations are primarily responsible. According to the research in (Horowitz 2014), a 32-bit floating-point (FL) consumes $4.6pJ$ for a MAC operation and $0.9pJ$ for an AC operation, namely $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$. Therefore, the theoretical energy consumptions of the l-th kernel point spiking layer and the l-th conventional kernel point convolution layer can be formulated as follows:

$$E^l_{P2SResLNet} = E^l_{kenel} + FLs^l_{P2SResLNet} \times E_{AC} \quad (15)$$

$$E^l_{KPConv} = E^l_{kenel} + FLs^l_{KPConv} \times E_{MAC} \quad (16)$$

Where E^l_{kenel} denotes the kernel computation energy consumption, which is currently based on floating-point operations. There is no difference for E^l_{kenel} between the two Equations. FLs^l_* represents the number of floating points in the feature computation of layer l. Assuming that in the l-th layer, the input number of 3D points is I; the number of neighboring points for each point is n; the size of each kernel weight is $D_{in} \times D_{out}$; F^{l-1}_r is the firing rate of layer $(l-1)$. We calculate FLs^l_* as follows:

$$FLs^l_{P2SResLNet} = I \times n \times D_{in} \times D_{out} \times F^{l-1}_r \quad (17)$$

$$FLs^l_{KPConv} = I \times n \times D_{in} \times D_{out} \quad (18)$$

According to the theoretical calculation above, we estimate the main energy consumptions of our network and KPConv (Thomas et al. 2019) regardless of point cloud downsampling and normalization operations in Figure 5, both with $6,597$ 3D points as inputs. Our network demonstrates great energy efficiency, over 30 times better than KPConv due to the sparse nature of spikes and the AC replacement.

Features Learned by KPS Neurons

To understand the learning mechanism of our network, we visualize the features learned by the KPS neurons in the third spiking residual block. We color the points based on their activation for some features in Figure 6. In the first row, points with activated 6-th feature are colored in red, while the rest are colored in blue. Here activated 6-th feature means the 6-th KPS neuron of a 3D point produce a spike at the time step $t = 1$. We can observe that red points are generally distributed on load-bearing plates, which indicate the 6-th KPS neuron in the third spiking residual block may extract the

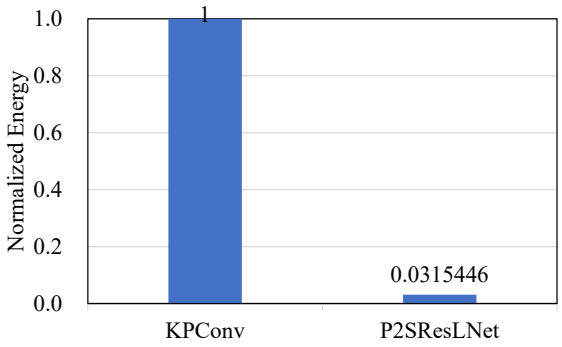

Figure 5: Normalized energy consumption comparison of KPConv and our network.

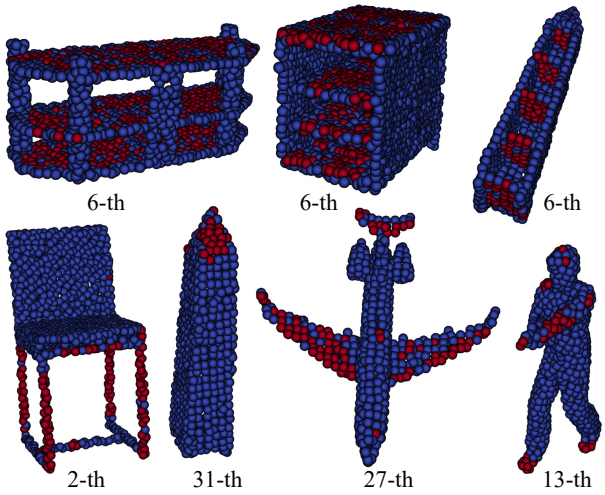

Figure 6: Features learned by KPS neurons in the third spiking residual block. The number below each object represents the dimension from which the extracted feature come from. We color an activation with a spike in red, and otherwise in blue. All activations are projected to the 3D input points.

load-bearing property of some objects. In the second row, we observe that the 2-th KPS neuron can learn the chair leg characteristics, the 31-th KPS neuron can learn the bookshelf top characteristics, the 27-th KPS neuron can learn the aircraft wing characteristics, the 13-th KPS neuron can learn the human limb characteristics.

Conclusion

In this paper, we propose the point-to-spike residual learning network for 3D point cloud classification that reaches a favourable balance between accuracy and bioinspired energy efficiency on two benchmark datasets, ModelNet40 and ScanObjectNN. In the future, we will focus on the deployment of 3D processing on neuromorphic chips and maintaining the high accuracy as some 3D processing SOTAs.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62206001, 62236002, 62102387, 62206005, the University Synergy Innovation Program of Anhui Province (GXXT-2022-041).

References

Bulsara, A. R.; Elston, T. C.; Doering, C. R.; Lowen, S. B.; and Lindenberg, K. 1996. Cooperative behavior in periodically driven noisy integrate-fire models of neuronal dynamics. *Physical Review E*, 53(4): 3958.

Cao, Y.; Chen, Y.; and Khosla, D. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113: 54–66.

Choy, C.; Gwak, J.; and Savarese, S. 2019. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3075–3084.

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1): 82–99.

Fang, W.; Chen, Y.; Ding, J.; Chen, D.; Yu, Z.; Zhou, H.; and Tian, Y. o. c. 2020. Spikingjelly.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; and Tian, Y. 2021. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34: 21056–21069.

Gerstner, W.; and Kistler, W. M. 2002. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Horowitz, M. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, 10–14. IEEE.

Kim, S.; Park, S.; Na, B.; and Yoon, S. 2020. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 11270–11277.

Lee, C.; Sarwar, S. S.; Panda, P.; Srinivasan, G.; and Roy, K. 2020. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, 119.

Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; and Chen, B. 2018. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31.

Ma, X.; Qin, C.; You, H.; Ran, H.; and Fu, Y. 2022. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. *arXiv preprint arXiv:2202.07123*.

Oikonomou, K. M.; Kansizoglou, I.; and Gasteratos, A. 2023. A hybrid spiking neural network reinforcement learning agent for energy-efficient object manipulation. *Machines*, 11(2): 162.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

Sun, Y.; Zeng, Y.; Zhao, F.; and Zhao, Z. 2023. Multi-compartment Neuron and Population Encoding improved Spiking Neural Network for Deep Distributional Reinforcement Learning. *arXiv preprint arXiv:2301.07275*.

Tatarchenko, M.; Park, J.; Koltun, V.; and Zhou, Q.-Y. 2018. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3887–3896.

Thomas, H.; Qi, C. R.; Deschaud, J.-E.; Marcotegui, B.; Goulette, F.; and Guibas, L. J. 2019. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6411–6420.

Uy, M. A.; Pham, Q.-H.; Hua, B.-S.; Nguyen, T.; and Yeung, S.-K. 2019. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1588–1597.

Wijaya, K. T.; Paek, D.-H.; and Kong, S.-H. 2022. Advanced feature learning on point clouds using multi-resolution features and learnable pooling. *arXiv preprint arXiv:2205.09962*.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

Zhang, D.; Zhang, T.; Jia, S.; and Xu, B. 2022. Multi-sacle dynamic coding improved spiking actor network for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 59–67.

Zhang, N.; Pan, Z.; Li, T. H.; Gao, W.; and Li, G. 2023. Improving Graph Representation for Point Cloud Segmentation via Attentive Filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1244–1254.

Zhou, C.; Yu, L.; Zhou, Z.; Zhang, H.; Ma, Z.; Zhou, H.; and Tian, Y. 2023. Spikingformer: Spike-driven Residual Learning for Transformer-based Spiking Neural Network. *arXiv preprint arXiv:2304.11954*.

Zhou, S.; Li, X.; Chen, Y.; Chandrasekaran, S. T.; and Sanyal, A. 2021. Temporal-coded deep spiking neural network with easy training and robust performance. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11143–11151.