Journal of Scientific Computing (2022) 90:39
https://doi.org/10.1007/s10915-021-01699-2

NPTC-net: Narrow-Band Parallel Transport Convolutional Neural Networks on Point Clouds

Pengfei Jin[1] · Tianhao Lai[1] · Rongjie Lai[2] · Bin Dong[1,3,4]

Received: 31 March 2021 / Revised: 20 August 2021 / Accepted: 27 September 2021 /
Published online: 8 December 2021

Abstract
Convolution plays a crucial role in various applications in signal and image processing, analysis, and recognition. It is also the main building block of convolution neural networks (CNNs). Designing appropriate convolution neural networks on manifold-structured point clouds can inherit and empower recent advances of CNNs to analyzing and processing point cloud data. However, one of the major challenges is to define a proper way to "sweep"filters through the point cloud as a natural generalization of the planar convolution and to reflect the point cloud's geometry at the same time. In this paper, we consider generalizing convolution by adapting parallel transport on the point cloud. Inspired by a triangulated surface-based method [46], we propose the Narrow-Band Parallel Transport Convolution (NPTC) using a specifically defined connection on a voxel-based narrow-band approximation of point cloud data. With that, we further propose a deep convolutional neural network based on NPTC (called NPTC-net) for point cloud classification and segmentation. Comprehensive experiments show that the proposed NPTC-net achieves similar or better results than current state-of-the-art methods on point cloud classification and segmentation.

R. Lai's work is supported in part by an NSF Career Award DMS-1752934. Bin Dong is supported in part by Beijing Natural Science Foundation (Z180001), NSFC 12090022 and Beijing Academy of Artificial Intelligence (BAAI).

✉ Rongjie Lai
lair@rpi.edu

✉ Bin Dong
dongbin@math.pku.edu.cn

Pengfei Jin
jinpf@pku.edu.cn

Tianhao Lai
howeverlth@pku.edu.cn

1 Beijing International Center for Mathematical Research, Peking University, Beijing, China

2 Department of Mathematics, Rensselaer Polytechnic Institute, Troy, NY, USA

3 Center for Data Science, Peking University, Beijing, China

4 Beijing Institute of Big Data Research, Beijing, China

Keywords Geometric deep learning · Computer vision · Parallel transport · Point cloud · Geometric convolution

Mathematics Subject Classification 68U05 · 65D18 · 68T45

1 Introduction

Data that arises from many applications in science and engineering is commonly represented in non-Euclidean structures, such as meshes, point clouds, and graphs. Such data includes 3D shapes in computer graphics, scanned point clouds in remote sensing, social networks in social science, functional networks in neuroscience [17,25,37,56], etc. These unstructured data often need to be properly analyzed first before they can be effectively utilized in various downstream tasks, such as classification, segmentation, registration, prediction, etc.

Recently, deep learning has enabled major breakthroughs in many fields in science and engineering. This is largely due to the capability of deep neural networks in extracting features, analyzes features, and make high-level decisions and predictions in an end-to-end fashion. Among different types of deep neural networks, the convolutional neural networks (CNNs) is particularly effective, especially for analyzing Euclidean data such as audios, images, and videos [13,27]. Due to the success of CNNs in analyzing Euclidean data, much effort has recently been made to extend CNNs to non-Euclidean data, which is one of the main objectives in geometric deep learning [6,8,35,36]. However, generalizing CNNs to non-Euclidean data is not straightforward. A typical CNN is a composition of simple operators such as convolution operators, down-sampling/pooling operators, batch normalization, etc. Notice that the convolution operator, which is the crucial operator in any CNNs, does not admit a simple extension to non-Euclidean data.

This article focuses on generalizing the convolution operator on (manifold structured) point clouds so that it inherits desirable properties of the planar convolution. This in turn enables us to design CNNs on point clouds. In the remaining part of this introduction, we shall first briefly review the applications of point cloud data, and provide a unified view on the existing extensions of convolutions on non-Euclidean data and discuss their relations with our proposed convolution, which is called the narrow-band parallel transport convolution.

1.1 Applications of Point Clouds Data

Point cloud data can be acquired by 3D laser scanners, such as Light Detection and Ranging (LIDAR) and RGB-D cameras. It can also be obtained by 3D scene reconstruction from 2D images, such as reconstructing the earth's landscape through aerial photography. Furthermore, one can obtain point clouds through the sampling of CAD models. Formally, a point cloud $\mathcal{P} = \{x_i \in \mathbb{R}^3 : i = 1, \ldots, N\} \subset \mathbb{R}^3$ consists of coordinates of points in a 3-dimensional Euclidean space. Unlike meshes and graphs, point clouds have no connectivity information.

Point clouds provide a direct and convenient way of representing geometric data. For example, in autonomous driving, accurate environment perception is needed to realize reliable navigation and decision in a complex dynamic environment [31]. Traditionally, image data can provide 2D semantic and texture information with low cost and high efficiency. However, image data lacks 3D geographic information. Therefore, dense and accurate point cloud data with 3D geographic information collected by LIDAR is commonly used by modern autonomous vehicles. In addition, LIDAR is insensitive to the change of lighting conditions

and can work both in the daytime nighttime. Other than autonomous driving, many applications prefer to use point cloud data, such as face recognition in e-commerce, urban planning and agricultural production, animation and virtual reality, etc. [32]. Due to the vast importance of point cloud data, we would like to extend the convolution operator and design CNNs on point clouds.

1.2 Related Work of Convolutions on Non-euclidean Domains

Convolution is one of the most widely used operators in applied mathematics, computer science, and engineering. It is also the most important building block of Convolutional Neural Networks (CNNs), which are the main driven force in the recent success of deep learning.

In the Euclidean space $\mathbb{R}^n$, the convolution of function f with a kernel (or filter) k is defined as

$$(f * k)(x) := \int_{\mathbb{R}^n} k(x-y) f(y) dy. \tag{1}$$

This operation can be easily calculated in Euclidean spaces due to the shift-invariance of the space so that the translates of the filter k, i.e., $k(x-y)$ is naturally defined.

One of the main challenges of proposing geometric meaningful convolution on manifolds and point clouds (a discrete form of manifolds) is to define an analogy of the Euclidean translation $x-y$ on the non-Euclidean domain. Multiple types of generalized convolutions on manifolds, graphs, and point clouds have been proposed in recent years. We shall review some of them and discuss the relation between existing definitions of convolutions and the proposed narrow-band parallel transport convolution.

Spectral methods avoid the direct definition of translation $x-y$ by utilizing the convolution theorem [20]: given any two functions f and g, $\widehat{f * g} = \hat{f} \cdot \hat{g}$. Therefore, we have $f * g = (\hat{f} \cdot \hat{g})^{\vee}$, where $\wedge$ and $\vee$ represent generalized Fourier transform and inverse Fourier transform provided through the associated Laplace-Beltrami (LB) eigensystem on manifolds. To avoid computing convolution through full eigenvalue decomposition of the LB operator, polynomial approximation has been proposed and yields convolution as the action of polynomials of the LB operator [10,15]. Thus, convolutional neural networks can be designed [7,9,28]. Spectral methods, however, suffer two major drawbacks. First, these methods define convolution in the frequency domain. As a result, the learned filters are not spatially localized. Second, spectral convolutions are domain-dependent as deformation of the ground manifold will change the corresponding LB eigensystem. This obstructs the use of learning networks from one training domain to a different testing domain [46].

Spatial mesh-based methods are more intuitive and similar to the Euclidean case, and this is one of the reasons why most of the existing works fall into this category [5,35,46]. The philosophy behind these methods is that the tangent plane $\mathcal{T}_x\mathcal{M}$ of a 2-dimensional manifold $\mathcal{M}$ is embedded to a 2-dimensional Euclidean domain where convolution can be easily defined. In this paper, we make the first attempt to interpret some of the existing mesh-based methods in a unified framework. We claim that most of the spatial mesh-based methods can be formulated as

$$(f * k)(x) := \int_{\mathcal{T}_{x,\epsilon}\mathcal{M}} k(\phi(x,\mathbf{v})) f(\mathbf{v}) \mathrm{d}\mathbf{v}, \quad x \in \mathcal{M}. \tag{2}$$

Here, $k : \mathbb{R}^2 \to \mathbb{R}$ is a convolution kernel and $\mathcal{T}_{x,\epsilon}\mathcal{M} = \{\mathbf{v} \in \mathcal{T}_x\mathcal{M} : \langle \mathbf{v}, \mathbf{v} \rangle_{g_x} \leq \epsilon^2\}$ with $\epsilon > 0$ being the size of the kernel. Given vector fields $\mathbf{u}^j$, $j = 1, 2$. The mapping

$\phi(x,\cdot) : \mathcal{T}_x\mathcal{M} \to \mathbb{R}^2$ is defined as

$$\phi(x,\mathbf{v}) = \left(\langle \mathbf{v}, \mathbf{u}_x^1\rangle_{g_x}, \langle \mathbf{v}, \mathbf{u}_x^2\rangle_{g_x}\right), \tag{3}$$

Most of the designs of the existing manifold convolutions focused on the designs of $\mathbf{u}^j$. We remark that possible singularities will lead to no convolution operation at those points. These are isolated points on a closed manifold and do not affect experiment results. More discussions will be provided in Sect. 3.1.2.

For example, GCNN [35] and ACNN [5] construct a local geodesic polar coordinate system on a manifold, formulating the convolution as

$$(f * k)(x) = \int k\left((\phi \circ \psi)(\theta, r)\right)(\mathcal{Q}_x f)(r,\theta)\mathrm{d}r\mathrm{d}\theta,$$

where $\mathcal{Q}$ is a local interpolation function with interpolation domain an isotropic disc for GCNN and an anisotropic ellipse for ACNN. A local geodesic polar coordinate system on a manifold can also be transformed to a 2-dimensional planar coordinate system on its tangent plane. Such transformation is the mapping ψ, which is defined by the inverse exponential map: $\mathbf{v} = exp^{-1}(z(\theta, r))$ with $z(\theta, r)$ being a point in the local geodesic polar coordinate system at $x \in \mathcal{M}$ with coordinates (θ, r). With this, we can easily interpret ACNN within the framework of (2). Indeed, ACNN essentially chooses $\mathbf{u}_x^j$ as the directions of the principal curvature at point x. For GCNN, on the other hand, it avoids choosing a specific vector field on the manifold by taking max-pooling among all possible directions of $\mathbf{u}_x^1$ at each point. Such definition of convolution, however, ignores the correspondence of the convolution kernels at different locations.

The newly proposed PTC [46] defines convolution directly on the manifold while using tangent planes to transport kernels by a properly chosen parallel transport. PTC can be equivalently cast into the form of (2) using the inverse exponential map, and implementation of the proposed parallel transported is realized through choosing specific vector fields $\{\mathbf{u}^j\}_{j=1,2}$ guided by the Eikonal equation for transforming vectors along geodesic curves on manifolds. Similarly, some definitions of convolution depend on particular frames. PFCNN [58] uses the optimized frame as $\{\mathbf{u}^j\}_{j=1,2}$. The objective function is based on the angles between adjacent frames. CGCNN [59] uses two different data-driven local reference frames as $\{\mathbf{u}^j\}_{j=1,2}$.

Spatial point-based methods have wider applications due to their weaker assumptions on the data structure.

There are mainly two types of point-based convolution. The first type is to combine the information of points directly. These methods can be formulated as

$$(f * k)(x_i) := \sum_{x_j \in \mathcal{N}(x_i)} k(x_i, x_j) f(x_j), \tag{4}$$

where $\mathcal{N}(x_i) \subset \mathcal{P}$ is a neighborhood of x_i and kernel k takes different forms in different methods. PointNet [41] is an early attempt to extract features on point cloud. PointNet is a network structure without convolution, or alternatively we can interpret the convolution defined by PointNet has the simplest kernel $k(x_i, x_j) = \delta(x_i, x_j)$ where δ is the Kronecker-Delta. Various later works attempt to improve PointNet by choosing different forms of the kernel k. For example, PointNet++ [42] introduces a max pooling among local points, i.e. choosing kernel k as an indicator function: $k(x_i, x_j) = I_{x_j = \arg\max_{z \in \mathcal{N}(x_i)} f(z)}$. PointCNN [30] chooses $k(x_i,\cdot) = \beta' A_{x_i}$ where $\beta \in \mathbb{R}^K$ and $A_{x_i} \in \mathbb{R}^{K\times K}$ are trainable variables with $K = |\mathcal{N}(x_i)|$. DGCNN [54] proposes an "edge convolution"that can be viewed as

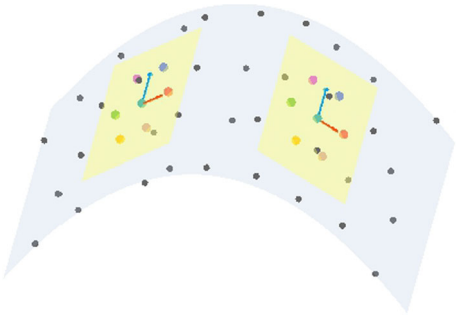

Fig. 1 Narrow-band parallel transport convolution on point cloud: black and blue points are sample points of the surface (blue). Each kernel (colored dots) is defined on the tangent plane (yellow). The vectors on the tangent planes that are of the same color are defined by parallel transport. (Color figure online)

fixing $f(x_j) \equiv 1$ in (4) and $k(x_i, x_j)$ =MLP($f(x_i)$, $f(x_i) - f(x_j)$), where MLP means the Multi-Layer Perceptron [40].

The second type of convolution is defined by first projecting the point cloud locally on a Euclidean domain and then employ regular convolution. This type of methods can also be formulate as (2). For example, Tangent convolutions [49] define kernels on the tangent plane, and use 2 principal directions of a local PCA as $\mathbf{u}_x$. Pointconv [55] constructs local kernels by interpolation in $\mathbb{R}^3$, i.e. letting $\phi(x, \mathbf{v}) = x - \mathbf{v}$ which is essentially a local Euclidean convolution. Similarly, [34,51,53] are also based on local Euclidean convolution. Their main differences are various forms of k and interpolation methods.

1.3 The Proposed Convolution: NPTC

We propose a Narrow-Band Parallel Transport Convolution (**NPTC**) in this paper. It is a geometric convolution based on point cloud discretization of a manifold parallel transport defined in a specific way. As we discussed in the previous section, convolutions in many methods can be written in the form of (2) and (3), while the differences mostly lie in the choices of the vector field $\{\mathbf{u}^j\}_j$. As observed by [46] that choosing the vector field properly, the associated convolution can be interpreted as transporting the kernels using the parallel transport associated with the prescribed vector field.

We attempt to define geometric convolutions that can be viewed as translating kernels on the point cloud in a parallel fashion. One naive approach to extend mesh-based methods to point cloud is to generate a triangulated surface based on the point cloud. However, this is not as convenient as working directly with the point cloud since in practice not every point cloud corresponds to a legitimate parameterized surface, and pooling is not as easy to implement on triangulated surfaces as on point clouds. In addition, it is time-consuming to construct meshes on point clouds. When applied in practice, mesh construction time may be much longer than the inference time of some methods directly applied on the point cloud. To avoid mesh construction and to handle point clouds data directly, we propose to define convolution on point clouds by combining voxelization and geometric convolution.

Now, we describe how NPTC is computed on point clouds. Firstly, point clouds are approximated by voxel-based narrow-band with appropriate resolution. For a manifold $\mathcal{M}$ or point cloud $\mathcal{P}$, its narrow-band is defined as the region with distance from $\mathcal{M}$ (or $\mathcal{P}$) less than ϵ: $\mathcal{NB}(\mathcal{M}) := \{x \in \mathbb{R}^3 | dist(x, \mathcal{M}) < \epsilon\}$, where $dist$ represents a distance function. For calculation efficiency and robustness to noise, we choose the voxel-based narrow-band approximation. It should be noted that the result of the traditional voxelization method [56] is a solid 3-d structure, and the voxel-based narrow-band can be understood as the "shell" with

a certain thickness of the former (if the point cloud is sampled from the surface of the object). It is very efficient to calculate the distance function in voxel approximation with appropriate resolution. The vector field $\{\mathbf{u}_x\}_{x\in\mathcal{M}}$ is defined as the projections of the gradient field of a narrow-band-based distance function on the approximated tangent plane of the point cloud. Such definition of the vector field is robust to noise, since if the distortion of the coordinates of the points by noise does not exceed the width of the narrow-band, the computed gradient field of the distance function in the narrow-band remains unchanged. After the vector field $\{\mathbf{u}_x\}_{x\in\mathcal{M}}$, we use local PCA to estimate normal vectors on the point cloud following [57]. Finally, the convolution kernel can be constructed on the tangent plane, and the corresponding $f(v)$ can be obtained by interpolating the value $f(x)$ on the point cloud.

Note that we prefer to use geometric convolution in NPTC because compared with methods that translate kernels in the ambient space of the manifold, NPTC translates kernels on the tangent planes, which effectively avoids having convolution kernels defined away from the underlying manifold of the point cloud. In other words, NPTC can well reflect point cloud geometry and is a natural generalization of planar convolution in the sense that when the point cloud reduces to planar grids, the NPTC reduces to the planar convolution.

1.4 Contributions

- We introduce a new point cloud convolution, NPTC, based on parallel transport defined by a narrow-band approximation of the point cloud. The proposed NPTC is a natural generalization of planar convolution.
- The proposed NPTC combines voxelization and geometric convolution. Voxelization with appropriate resolution brings robustness and geometric convolution can better reflect the point cloud's geometry.
- Based on NPTC, we designed convolutional neural networks, called NPTC-net, for point clouds classification and segmentation with state-of-the-art performance.

The rest of this paper is organized as follows. In Sect. 2, we will discuss the mathematical background of parallel transports on manifolds and the Eikonal equation for computing distance functions. In Sect. 3, we propose the narrow-banded parallel transport convolution, and it is associated with convolutional neural networks on point cloud represented manifolds. After that, we report our intensive numerical experiments of point clouds classification and segmentation on benchmark data sets in Sect. 4. We conclude the paper in Sect.5.

2 Background

2.1 Manifold and Parallel Transport

Let $\mathcal{M}$ be a two-dimensional differential manifold embedded in $\mathbb{R}^3$. We write $\mathcal{T}_x\mathcal{M}$ as the two-dimensional tangent plane at point $x \in \mathcal{M}$. The disjoint union of the tangent planes at each point on the manifold defines the tangent bundle $\mathcal{T}\mathcal{M}$. A vector field V is a smooth assignment: $\mathcal{M} \to \mathcal{T}\mathcal{M}$ such that $\mathbf{v}_x \in \mathcal{T}_x\mathcal{M}, \forall x \in \mathcal{M}$. The collection of all smooth vector fields is denoted as $\Gamma(\mathcal{T}\mathcal{M})$.

An affine connection is a bilinear mapping ∇: $\Gamma(\mathcal{TM}) \times \Gamma(\mathcal{TM}) \to \Gamma(\mathcal{TM})$, such that for all smooth functions f and g in $C^\infty(\mathcal{M})$ and all vector fields U, V and W on M:

$$\begin{cases} \nabla_{fU+gV} W = f\nabla_U W + g\nabla_V W, \\ \nabla_U(aV + bW) = a\nabla_U V + b\nabla_U W, \quad a, b \in \mathbb{R}, \\ \nabla_U(fV) = df(U)V + f\nabla_U V. \end{cases} \tag{5}$$

A vector field U is called **parallel** along a curve $\gamma : I \to \mathcal{M}$ if $\nabla_{\dot\gamma} U = 0$. Given an vector $\mathbf{e} \in \mathcal{T}_{x_0}\mathcal{M}$ at $x_0 = \gamma(0) \in \mathcal{M}$, the **parallel transport** of $\mathbf{e}$ along γ is the extension of $\mathbf{e}$ to a parallel section U on γ. More precisely, U is the unique section of $\Gamma(\mathcal{TM})$ along γ satisfying the ordinary differential equation $\nabla_{\dot\gamma(t)} U(t) = 0$ with the initial value $U(0) = \mathbf{e}$.

In differential geometry, a geodesic on a smooth manifold $\mathcal{M}$ with an affine connection ∇ is a curve $\gamma(t)$ such that parallel transport along the curve preserves the tangent vector to the curve. It is a generalization of the notion of a "straight line". Formally, a geodesic is $\gamma : [0, l] \to \mathcal{M}$ if $\nabla_{\dot\gamma(t)}\dot\gamma(t) = 0$. For any two points x_0 and x_1 on a closed manifold $\mathcal{M}$, there will be a geodesic connecting x_0 and x_1. More details on aforementioned concepts can be referred in [24].

Alternatively, an affine connection can be defined by an assignment Ξ as a family of linear transformations on tangent spaces along any smooth curve on $\mathcal{M}$. Consider γ_x^y a smooth arc joining two, not necessarily distinct, points from x to y, define $\Xi(\gamma_x^y) : \mathcal{T}_x\mathcal{M} \to \mathcal{T}_y\mathcal{M}$. If $\Xi(\gamma_x^y)$ satisfies the following properties:

1) $\Xi(\gamma_x^y)$ is non-singular,
2) $\lim_{y\to x} \Xi(\gamma_x^y) = Id$,
3) $\Xi(\gamma_x^z) = \Xi(\gamma_y^z)\Xi(\gamma_x^y)$,
4) Ξ is Frechét differentiable in terms of γ, x and y.

Then, the vector $\mathbf{v}_y$ is obtained by parallel displacement of $\mathbf{v}_x$ along γ_x^y is provided as $\mathbf{v}_y = \Xi(\gamma_x^y)\mathbf{v}_x$. Consider a tangent vector field V along γ, the associated infinitesimal connection $\nabla_{\dot\gamma} V = \lim_{h\to 0} \frac{1}{h}(\Xi(\gamma_{\gamma(h)}^{\gamma(0)})V_{\gamma(h)} - V_{\gamma(0)})$ can be induced from Ξ [23]. Thus, defining linear transformations on any arc satisfying specific properties is equivalent to defining connection and parallel transport.

In applications, we only need to transport kernels from fixed x_0 to any x instead of transformations along all arcs. In PTC [46], for any point x, let $\gamma_{x_0}^x$ be the geodesic curve connecting x and x_0. Define a linear transformation $\Xi(\gamma_{x_0}^x)$ satisfying $\mathbf{u}_x^j = \Xi(\gamma_{x_0}^x)\mathbf{u}_{x_0}^j$, $j = 1, 2$, where $\mathbf{u}_x^1$ is the geodesic curve's tangent vector at x and $\mathbf{u}_x^2 = \mathbf{u}_x^1 \times \mathbf{n}_x$. Then $\mathbf{v}_x = \Xi(\gamma_{x_0}^x)\mathbf{v}_{x_0}$ representing the parallel transport of $\mathbf{v}_{x_0}$ from x_0 to x along $\gamma_{x_0}^x$.

It is easy to check that $\phi(x, \mathbf{v}_x) = \phi(x_0, \mathbf{v}_{x_0})$, where $\phi(x, \mathbf{v})$ is defined in (3). For convenience, instead of transporting the kernel on the manifold, the parallel transported kernels can be locally constructed at every point x by formulating k as $k(\phi(x, \cdot))$.

2.2 Solving Eikonal Equation on Point Clouds

As mentioned before, the proposed NPTC relies on the computation of a distance function on a voxel-based narrow-band approximation of the given point cloud. Therefore, we briefly review what a narrow-band approximation is and how the distance function is calculated.

Geometric attributes calculation of large unstructured point-based data sets is a challenging task. Hence, the most common way of dealing with unstructured point-based data is to re-sample to a structured grid [12]. A large variety of well-known methods like isosurface extraction [33], region-growing methods [18], and level-set methods [38,39] can be applied to

gridded data. The original idea of level sets is to implicitly represent a surface as the solution of an equation concerning an underlying scalar field. To improve computation efficiency of level-set methods, local level-set method was introduced [2] which essentially suggests conduct computations only within a narrow-band of the zero level set.

Different methods are proposed to obtain narrow-band representation of shapes in 2D [4,19] or 3D [44,45]. In this work, the narrow-band of a point cloud $\mathcal{P}$ is defined as the region with distance from $\mathcal{P}$ less than ϵ: $\mathcal{NB}(\mathcal{P}) := \{x \in \mathbb{R}^3 | dist(x, \mathcal{P}) < \epsilon\}$, where $dist$ represents the standard Euclidean distance function.

We follow the idea of narrow-band representation and construct the voxel-based narrow-band. Consider the point cloud with N points in the unit cube and divide the unit cube into M^3 small cubes. The side length of each small cube is $\frac{1}{M}$. The set of all small cubes whose distance from the point cloud is less than ϵ forms the voxel-based narrow-band approximation of the point cloud. The number of grid points in traditional voxelization is M^3, while the number of grid points in the voxel-based narrow-band for a smooth surface is $O(K_{\mathrm{NB}} M^2)$, where $K_{\mathrm{NB}} = \lceil 2\epsilon M \rceil$ depends on the thickness of the narrow-band.

Distance functions can be easily computed by solving the Eikonal equation [11]. The Eikonal equation is a non-linear partial differential equation describing wave propagation:

$$|\nabla \rho| = 1/h(x), \quad x \in \Omega, \quad \rho|_\Lambda = 0, \tag{6}$$

where $\Lambda \subset \overline{\Omega} \subset \mathbb{R}^n$ and $h(x)$ is a strictly positive function. The solution $\rho(x)$ of (6) can be viewed as the shortest time needed to travel from x to Λ with $h(x)$ being the speed of the wave at x. For the special case when $h = 1$, the solution $\rho(x)$ represents the distance from x to Λ limited in the Ω. The Eikonal equation can be solved by the fast marching method [47] or the fast sweeping method [62]. Both methods have an optimal computation complexity of $O(L)$ with L being the number of the grid points. Naturally, when $N \gg M^2$, solving the equation in the voxel-based narrow-band will be faster than calculating directly on the point cloud or the classical voxelization. In fact, in practical applications, such as LiDAR data, the number of points in a point cloud is often far greater than the resolution required to adequately represent the shape of interest.

3 Narrow-band Parallel Transport Convolution (NPTC) and Network Design

Generalization of convolution defined by parallel transport on triangulated surfaces has already been proposed in [46]. In this section, we discuss how to transport kernels on point clouds in a similar fashion.

3.1 Narrow-band Parallel Transport Convolution (NPTC)

For a given function $f : \mathcal{P} \to \mathbb{R}$, the NPTC of f with kernel k takes the same form as (2). Under such formulation, the key to designing a convolution is to design vector fields $\{\mathbf{u}^j\}_{j=1,2}$. In this subsection, we discuss the general idea of NPTC and the interpretation of it in terms of parallel transport.

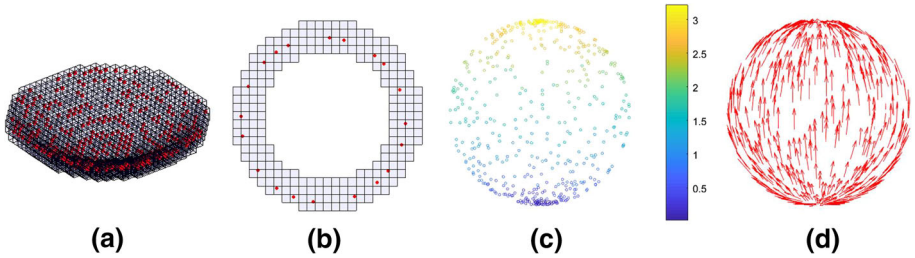

Fig. 2 Illustration of a point cloud $\mathcal{P}$ sampled from the unit sphere. **a** shows the narrow-band approximation (blue boxes) of part of $\mathcal{P}$ (in red). **b** is a cross section of (**a**). **c**, **d** show the distance function ρ and vector field $\{\mathbf{u}_x^1\}$ ($\{\nabla_{\mathcal{P}}\rho(x)\}$) on the point cloud. We can see that distance propagates from the bottom center to the top center reflecting the geometry of the sphere. (Color figure online)

3.1.1 General Idea of NPTC

To select a suitable vector field, we first recall the choice of the vector fields of PTC, which defines convolution on triangulated surfaces via parallel transport with respect to the Levi-Civita connection [46]. The geodesic curve represents the shortest path between two points on a Riemannian manifold. Given a geodesic connecting two points x and y, the tangential direction at x corresponds to the ascent direction of geodesic distance from y. PTC chooses such direction as $\mathbf{u}_x^1$ and defines $\mathbf{u}_x^2 = \mathbf{u}_x^1 \times \mathbf{n}_x$ where $\mathbf{n}_x$ is the normal vector at x.

To construct a vector field on a point cloud, we also consider using the gradient of a distance function on the point cloud. However, unlike triangulated surfaces, the distance function is not easily defined on point clouds due to the lack of connectivity. It is then natural to approximate the point cloud with another data structure with connectivity, so that the distance function can be easily calculated. For convenience and efficiency, we use voxelization [56] to approximate the point cloud in a narrow-band in $\mathbb{R}^3$ covering the point cloud. We denote such distance function as $\rho : \mathbb{R}^3 \to \mathbb{R}^+$. We will elaborate on how ρ can be calculated in later parts of this subsection.

Note that if the point cloud is sampled from a plane, the narrow-band is flat as well. Then, by a proper choice of the distance function, the vector fields $\{\mathbf{u}^j\}_{j=1,2}$, can be reduced to the global coordinate $\{\mathbf{e}_j\}$, $j = 1, 2$ on the plane. This means that NPTC is reduced to the traditional planar convolution.

Once the distance function ρ is computed, we choose $\mathbf{u}_x^1 = \nabla_{\mathcal{P}}\rho(x)$, where $\nabla_{\mathcal{P}}\rho(x)$ is a projection of $\nabla\rho(x)$ on an approximated tangent plane at x. Then, $\mathbf{u}_x^2$ can be calculated by the outer product $\mathbf{u}_x^2 = \mathbf{u}_x^1 \times \mathbf{n}_x$ with $\mathbf{n}_x$ the normal vector at x. The value $f(v)$ is computed by nearest-neighbor interpolation [14], i.e., $f(v) = f(z)$ where $z \in \mathcal{P}$ is the closest point to v. Note that one may use a more sophisticated method to compute $f(v)$ rather than using nearest-neighbor interpolation. We choose nearest-neighbor interpolation because of its simplicity.

3.1.2 Computing Distance Function on Point Clouds

A point cloud is entirely discrete without inherent connectivity. Therefore, it is not straightforward to compute the distance function on point clouds, although the local mesh method [26] can be applied to solve the Eikonal equation. For simplicity, we use voxels to approximate point clouds and to compute distance functions on the voxels using the well-known fast

marching method based on the regular grid provided by the voxelization [47]. Note that using voxels to compute distance functions is fast and robust to noise and local deformations.

The solution $\rho(x)$ of the Eikonal equation $|\nabla\rho(x)| = 1$ presents the distance form Λ to x limited inside the narrow-band. Here Λ is chosen as a certain point on the point cloud. We note that the starting point will become a singularity, but on non-parallelizable manifolds such as the sphere, it is not possible to construct a smooth vector field [16]. So that the vector field must have at least one a singularity. Although generating multiple vector fields by selecting different starting points and using pooling is helpful to eliminate singularities [46], experiments show that directly selecting one point already provides satisfactory results. Also, the ensemble result of several vector fields has almost no improvement compared with each of a single vector field.

Finally, we interpolate the distance function from the voxels to the point cloud.

3.1.3 Computing the Vector Fields on Point Clouds

We first compute the tangent plane on each point. Tangent planes are important features of manifolds and have been well-studied in the literature [26]. In this paper, we use local principal component analysis (LPCA) to estimate the tangent plane. We estimate the local linear structure near $x \in \mathcal{P}$ using the covariance matrix

$$\sum_{x_i \in \mathcal{N}(x)} (x_i - c)^\top (x_i - c), \quad c = \frac{1}{|\mathcal{N}(x)|} \sum_{x_i \in \mathcal{N}(x)} x_i,$$

where $\mathcal{N}(x)$ is the set of neighboring points of x. The eigenvectors of the covariance matrix form an orthogonal basis. If the point cloud is sampled from a two dimensional manifold, and the local sampling is dense enough to resolve local features, the eigenvectors corresponding to the largest two eigenvalues provide the two orthogonal directions of the tangent plane, and the remaining vector represents the normal direction at $x \in \mathcal{P}$. Here, we denote the space spanned by the two eigenvectors of the covariance matrix at x as $\mathcal{T}_x\mathcal{P} \subset \mathbb{R}^3$.

With the computed distance function $\rho(x)$, it is nature to define the vector field by projecting $\nabla\rho$ on the approximated tangent planes of the point cloud. Given a point $x_i \in \mathcal{P}$ close enough to $x \in \mathcal{P}$, we have

$$\langle \nabla\rho, x_i - x \rangle \approx \rho(x_i) - \rho(x),$$

where $\rho(x_i)$ and $\rho(x)$ are known. If we consider K-nearest neighbors of x, we have $K - 1$ equations with 3 unknowns that are the three components of $\nabla\rho$. We can use least squares to find $\nabla\rho$. We then project the vector $\nabla\rho(x)$ onto the tangent plane at x. We denote the projected vector $\nabla_{\mathcal{P}}\rho$, which is the vector we eventually need to define NPTC as described in Sect. 3.1.1. Denote the calculated vector field as $\{\mathbf{u}^j\}_{j=1,2}$, and the starting point of the distance function as x_0. For any point x, let $\gamma_{x_0}^x$ be the integral curve of $\nabla\rho$ connecting x and x_0. Define a linear transformation $\Xi(\gamma_{x_0}^x)$ satisfying $\mathbf{u}_x^j = \Xi(\gamma_{x_0}^x)\mathbf{u}_{x_0}^j$, $j = 1, 2$. As mentioned in the Sect. 2, defining linear transformations on any arc satisfying specific properties means defining connection and parallel transport. In applications, we do not need linear transformations along all arcs. NPTC is defining linear transformations on integral curves of $\nabla\rho$ (as approximated geodesic curves).

However, the quality of vector fields is affected by the quality of the estimated normal directions. One natural concern is how such estimation is affected by noise. We remark that LPCA is in fact quite robust to noise. In the section of experiments, we will observe that our method can still achieve satisfactory results on the real-world scanned dataset with noise.

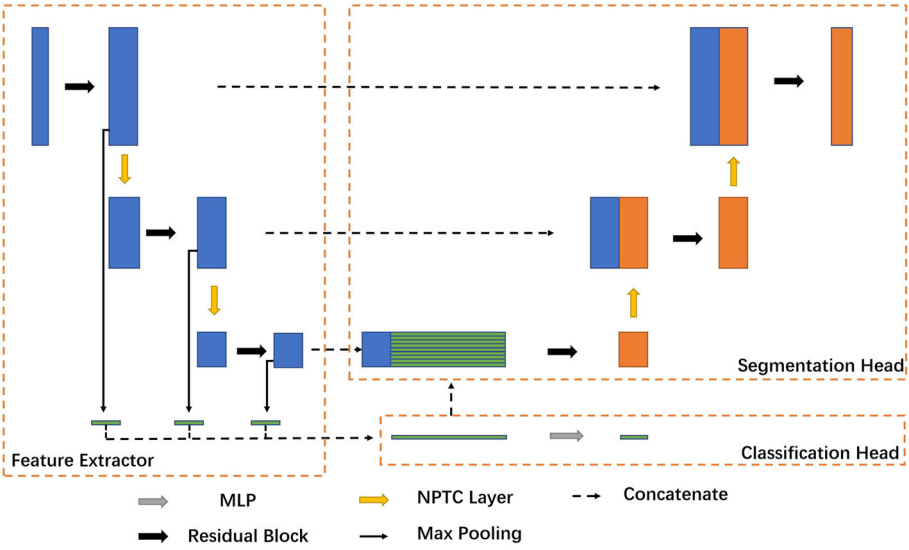

Fig. 3 Architecture of NPTC-net. The network for segmentation tasks is at the top right with encoding and the network for classification tasks is at the bottom right with encoding.

3.2 NPTC-net: Architecture Design for Classification and Segmentation

This section, we present how to use NPTC to design convolutional neural networks on point clouds for classification and segmentation tasks. For that, other than the NPTC, we need to define some other operations that are frequently used in neural networks.

Down-sampling Given the input point cloud $\mathcal{P}$, we use farthest point sampling [11] to generate a sequence $\{\mathcal{P}^{(i)}\}$, where $\mathcal{P}^{(0)} = \mathcal{P}$ and $\mathcal{P}^{(i)} \subset \mathcal{P}^{(i-1)}$ represents the i-th hierarchical structure of $\mathcal{P}$. The down-sampled set of $\mathcal{P}^{(i)}$ is $\mathcal{P}^{(i+1)}$ and up-sampled set is $\mathcal{P}^{(i-1)}$ during convolution layer. In i-th layer, feature maps is denoted as $\mathcal{F}^{(i)} \in \mathbb{R}^{N^{(i)} \times c^{(i)}}$, where $N^{(i)}$ is the number of the points and $c^{(i)}$ is the number of channels at layer i.

Multi-Layer Perceptron(MLP) Given the feature maps $\mathcal{F}$, MLP($\mathcal{F}$) is defined as $\sigma(...\sigma(\mathcal{F}W^{(1)})W^{(2)}...)W^{(L)})$, where σ is a non-linear function and $W^{(i)}$ is a matrix.

Concatenate Given the feature maps $\mathcal{F}_1 \in \mathbb{R}^{N \times c_1}$ and $\mathcal{F}_2 \in \mathbb{R}^{N \times c_1}$, concatenate of $\mathcal{F}_1$ and $\mathcal{F}_2$ is defined as $\mathcal{F}_3 = [\mathcal{F}_1; \mathcal{F}_2] \in \mathbb{R}^{N \times (c_1 + c_2)}$.

Global max pooling Given the feature maps $\mathcal{F} \in \mathbb{R}^{N \times c}$, $\hat{f} \in \mathbb{R}^c$ is the global max pooling of $\mathcal{F}$ satisfying $\hat{f}_j = \max_i \mathcal{F}_{ij}$.

NPTC layer Our i-th NPTC layer takes points $\mathcal{P}^{(i)} \in \mathbb{R}^{N^{(i)} \times 3}$ and their corresponding feature maps $\mathcal{F}^{(i)} \in \mathbb{R}^{N^{(i)} \times c^{(i)}}$ as input. The corresponding output is $\mathcal{F}^{(i+1)} \in \mathbb{R}^{N^{(i+1)} \times c^{(i+1)}}$ living on the points $\mathcal{P}^{(i+1)} \in \mathbb{R}^{N^{(i+1)} \times 3}$. The NPTC-net have encoding and decoding stages. Normally, $N^{(i+1)} \leq N^{(i)}$ during encoding and $N^{(i+1)} \geq N^{(i)}$ during decoding. Convolution at the i-th layer during encoding is only performed on the point set $\mathcal{P}^{(i+1)}$, which resembles convolution with stride ≥ 1 for planar convolutions.

Residual block One residual block takes the feature maps $\mathcal{F} \in \mathbb{R}^{N \times c}$ on the point set $\mathcal{P} \in \mathbb{R}^{N \times 3}$ as input and the same number of points and the same number of channels of features as output. One residual layer consists of three components: MLP from c channels to $\frac{c}{2}$ channels, convolution layer from $\frac{c}{2}$ channels to $\frac{c}{2}$ channels, MLP from $\frac{c}{2}$ channels to

c channels plus the feature maps from the bypass connection. A residual block consists of several residual layers.

NPTC-net consists of the aforementioned operations and its architecture is given by Fig. 3. The left half of the NPTC-net is the encoder part of the network for feature extraction. For classification, features at the bottom of the network are directly attached to a classification network; while for segmentation, features are decoded using the right half of the NPTC-net (decoder part of the network) to output the segmentation map. Each rectangle represents a feature map, the height of the rectangle represents the number of points, and the width represents the number of channels.

4 Experiments

In order to evaluate our new NPTC-nets, we conduct experiments on three widely used 3D datasets, ModelNet [56], ShapeNet Part [60], S3DIS [3]. ModelNet40 and ShapeNet Part contain point clouds generated from CAD models, S3DIS is a real-world scanned dataset. ModelNet40 is for 3D shape classification, and ShapeNet Part and S3DIS are for segmentation (or partition) of 3D shapes.

4.1 Implementation Details

We implement the model with Tensorflow [1]. The neural network is denoted as f_θ, where θ corresponds to the trainable parameters of the network. Given labeled data pair $\{P_i, y_i\}$, where P_i corresponds to the point cloud data (i.e. the collection of 3D points of a given geometric object) and y_i is its associated label, the learning tasks is formulated as the optimization problem $\min_\theta \sum_i L(f_\theta(P_i), y_i)$ for some loss function L. For classification, L is the cross-entropy loss: $L(f_i, y_i) = -\sum_{j=1}^{c} y_{ij} log(f_{ij})$, where y_{ij} corresponds to the j-th element of one-hot encoded label of y_i and f_{ij} denotes the j-th element of f_i. Segmentation task can be viewed as point-wise classification task with cross-entropy loss.

We using SGD optimizer [43] with an initial learning rate 0.1 for ModelNet40 and ADAM optimizer [21] with an initial learning rate 0.002 for ShapeNet Part and S3DIS on a GTX TITAN Xp GPU.

To avoid over-fitting, data augmentation [48] is used during training. Data augmentation is to generate more data pairs $\{\mathcal{A}_j(P_i), y_i\}$, where $\mathcal{A}_j$ corresponds to some transformation without changing the label. For the classification task, the data is augmented by random rotation, scaling, and Gaussian perturbation on the coordinates of the points. For segmentation, the data is augmented by scaling and Gaussian perturbation on the coordinates of the points.

We do inferences on the augmented data during testing and aggregate the results by voting following [42]. Let $\hat{y}_{ij} = f_\theta(\mathcal{A}_j(P_i))$ denote the prediction on $\mathcal{A}_j(P_i)$. The aggregated inference result of P_i is $\hat{y}_i = \frac{1}{n_a}\sum_{j=1}^{n_a} \hat{y}_{ij}$.

Following [30], the data of S3DIS is firstly split by room, and then the rooms are sliced into $1.5m$ by $1.5m$ blocks, with $0.3m$ padding on each side. The points in the padding areas serve as context of the internal points, and themselves are not linked to loss in the training phase, nor used for prediction in the testing phase. Each block will be viewed as a point cloud during training and testing.

blueIn the following experiments, we choose the neighborhood $|\mathcal{N}(x)| = 20$ when using LPCA and least squares to calculate the gradient of the distance function. The narrow-

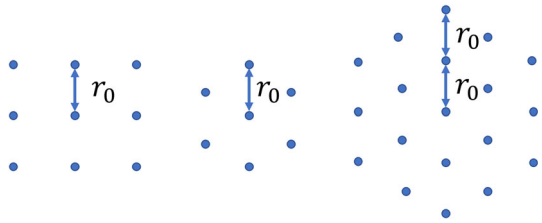

Fig. 4 Three different types of convolutional kernels: 3×3 rectangular kernel, single-ring hexagonal kernel, and double-ring hexagonal kernel.

Table 1 Comparisons of overall accuracy (OA) and mean per-class accuracy (mA) on ModelNet40 as well as comparisons in instance average IoU (mIoU) and class average IoU (mcIoU) on ShapeNet Part.

	ModelNet40		ShapeNet part	
Method	OA(%)	mA(%)	mIoU	mcIoU
kd-net [22]	91.8	**88.5**	82.3	77.4
PointNet [41]	89.2	86.2	83.7	80.4
SO-net [29]	90.9	87.3	84.9	81.0
PointNet++ [42]	90.7	–	85.1	81.9
SpecGCN [52]	92.1	–	85.4	–
SpiderCNN [57]	**92.4**	–	85.3	81.7
PointCNN [30]	92.2	88.1	***86.1***	***84.6***
Ours	***92.7***	***90.2***	**85.8**	**83.3**

Models ranking first is in bold italic and second is in bold.

band is constructed with the resolution of voxelization $M = 100$ and the number of voxels corresponds to the thickness $K_{\mathrm{NB}} = \lceil 2\epsilon M \rceil = 2$.

In our experiments, we have tried three shapes of convolutional kernels with different supports in the experiments shown in the following figure, where r_0 is a hyper-parameter. Taking the experiments on ModelNet40 classification as an example, the accuracy using the second shape of kernels is 0.6% higher than that using the first shape on average. The classification using the second and third shapes of kernels have very similar accuracy, while the third one is more computationally intensive. Therefore, we select the second shape of the convolution kernel for all experiments.

4.2 3D Shape Classification and Segmentation

We test the NPTC-net on ModelNet40 for classification tasks. ModelNet40 contains 12,311 CAD models from 40 categories with 9,842 samples for training and 2,468 samples for testing. For comparison, we use the data provided by [41] sampling 2,048 points uniformly and computing the normal vectors from the mesh. As shown on Table 1, our networks outperform other state-of-art methods. (If a compared method has results on both 2048 (or 1024) and 5000 points, we only compare with the former.)

We also evaluate the NPTC-net on ShapeNet Part for segmentation tasks. It contains 16,680 models from 16 shape categories with 14,006 for training and 2,874 for testing, each annotated with 2 to 6 parts, and there are 50 different parts in total. We follow the experiment setup of previous works, putting object categories into networks as known information. We use point intersection-over-union (IoU) to evaluate our NPTC-net. Table 1 shows that our model ranks second on this dataset and is fairly close to the best-known result.

Table 2 Comparisons of overall accuracy (OA) and mean per-class IoU (mIoU) on S3DIS.

Convolution Type	Method	OA(%)	mIoU(%)
No convolution	PointNet [41]	78.8	41.3
3-d convolution	SegCloud [50]	–	48.9
	Eff3DConv[61]	69.3	51.8
	ParamConv[53]	–	***58.3***
Geometric convolution	TangentConv [57]	**82.5**	52.8
	Ours	***83.7***	**54.0**

Models ranking first is in bold italic and second is in bold.

4.3 3D Scene Semantics Segmentation

As pointed out earlier, using voxelization can bring robustness to the definition of geometric convolution. In order to show the robustness of our model for real data, we tested scene semantics segmentation on the "Stanford Large-Scale 3D Indoor Spaces Dataset" (S3DIS). S3DIS covers six large-scale indoor areas from 3 different buildings for a total of 273 million points annotated with 13 classes. This is a real-world scanned dataset without normal and with noise. Following [50], we advocate the use of Area-5 as test scene to better measure the generalization ability of our method. Table 2 shows that geometric convolution methods are close to the methods which do not need normal estimation. NPTC-net outperforms the best known geometric convolution method TangentConv[57].

4.4 Visualization of Segmentation

To better demonstrate the advantages of our model, we visualize the segmentation results of the test data in the ShapeNet Part. We compare NPTC-net with the popular model PointNet++ [42] and select several representative examples. In Figs. 5, 6, three columns from left to right represent the ground truth, the prediction by PointNet++, and the prediction by NPTC-net. Each blue box of Fig. 5 contains objects within the same class (handgun, chair, and skateboard). In each blue box, the first row contains objects with standard structures, while the second row contains objects with certain unusual structures, which is highlighted by red boxes. In each blue box of Fig. 6, the first row contains the overviews of the objects while the second row contains the corresponding zoom-in views.

As shown in Fig. 5, we found that both models can make good predictions on objects with standard structures. However, if the object contains some unusual structures, such as abnormal shape, tilt, or asymmetry, NPTC-net is able to generate smoother results that respect geometric information.

4.5 Running Statistics

As shown in Table 3, we summarize our running statistics based on the model for ModelNet40 with batch size 16. In comparison with several other methods, although we use ResNet structure, the fewer channels, smaller kernels, and simpler interpolation (nearest neighboring) make NPTC use similar parameters and even fewer FLOPs.

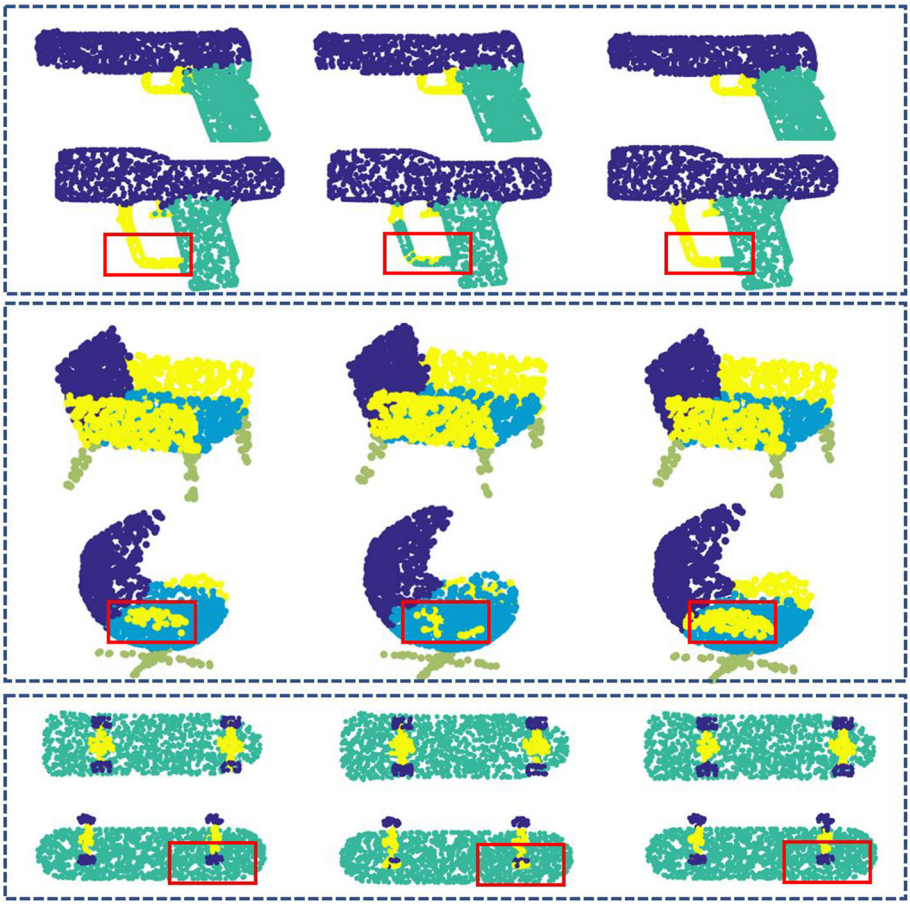

Fig. 5 Comparison of segmentation results of unusual structures: three columns from left to right represent the label, the prediction of PointNet++, and the prediction of NPTC-net.

Table 3 Comparisons of number of parameters and FLOPs for classification.

method	Parameters	FLOPs(Inference)
PointNet [41]	3.48M	14.70B
PointNet++ [42]	1.48M	26.94B
PointCNN [30]	**0.6M**	25.30B
Ours	1.29M	**11.7B**

Models with the least amount of parameters or the least FLOPs are bolded

Total pre-processing time mainly depends on the grid's density. For most cases, the resolution of 100^3 is enough to describe the shape of the point cloud, which is what we chose for our experiments. We also remark that the whole computation cost of constructing convolution depends linearly on the resolution of the voxel and the size of the data. On a PC with Core i7-7700 CPU, the pre-processing takes about **0.5** seconds per point cloud with Matlab, and the whole dataset of ModelNet40 (12,311 shapes with 10,000 points each) takes only

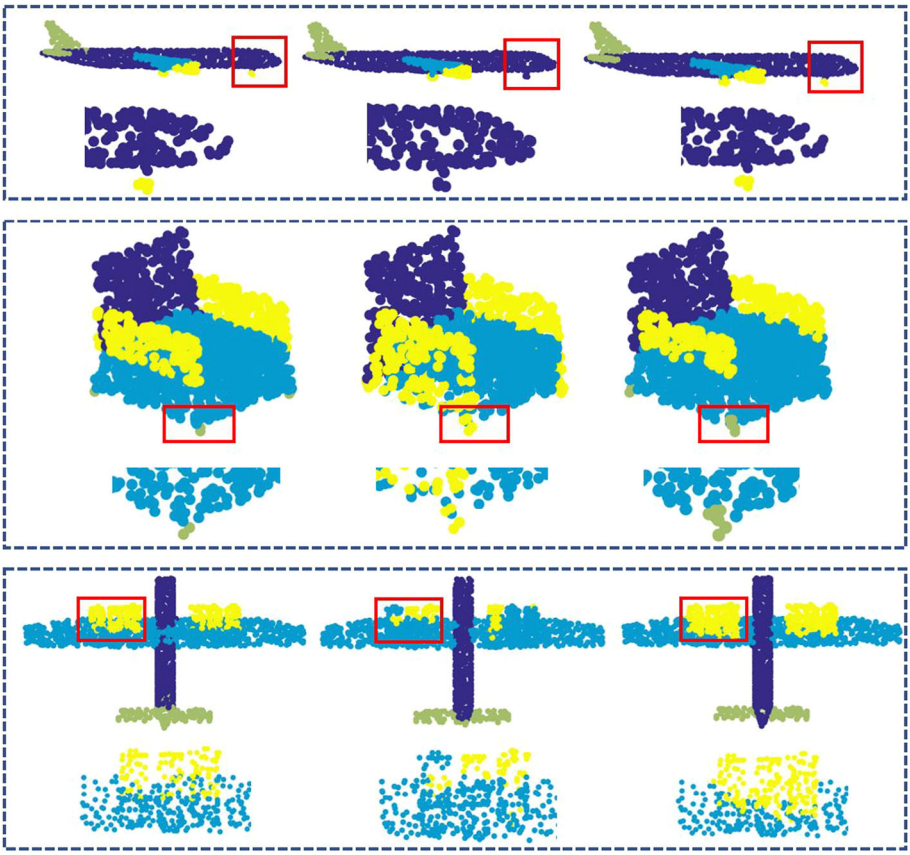

Fig. 6 Comparison of segmentation results of small structures: three columns from left to right represent the label, the prediction of PointNet++, and the prediction of NPTC-net.

Table 4 Comparisons of training time of networks on ModelNet40

Method	Settings	Accuracy and Training (+Pre-processing) time
PointNet++ [42]	adam, 1024 points	90.7%, 6 hours
	adam, 5000 points	91.9%, 20 hours
Ours	adam, 2048 points	92.4%, 6h (+1.5h)
	SGD, 2048 points	92.7%, 12h (+1.5h)

about 1.5 hours. It is negligible compared to the training of the deep neural networks and acceptable to make inferences in practice.

4.6 Feature Visualization

To visualize the effects of the proposed NPTC in the NPTC-net, we trained the network on ShapeNet Part and visualized learned features by coloring the points according to their level of activation. In Fig. 7, filters from the first Convolution layer in the first Residual block

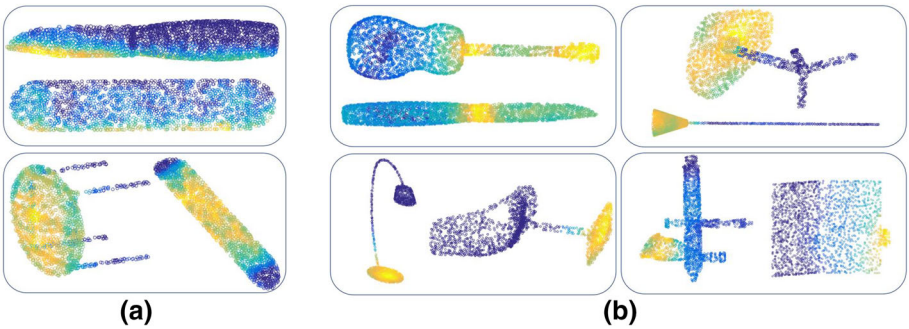

Fig. 7 Feature Visualization: each feature from low (**a**) or high (**b**) level is displayed on 2 point clouds from different categories. High-level activations are in yellow and low-level activations are in blue. (Color figure online)

and the final Convolution layer in the second Residual block are chosen. In order to easily compare the features at different levels, we interpolate them on the input point cloud. Observe that low-level features mostly represent simple structures like edges (top of (A)) and planes (bottom of (A)) with low variation in their magnitudes. In deeper layers, features are richer and more distinct from each other, like bottleneck (upper left of (B)), "big-head"(upper right of (B)), plane base (lower left of (B)), bulge (lower right of (B)).

5 Conclusion

This paper proposed a new way of defining convolution on point clouds, called the narrow-band parallel transport convolution (NPTC), based on a point cloud discretization of a manifold parallel transport. The parallel transport was defined specifically by a vector field generated by the gradient field of a distance function on a narrow-band approximation of the point cloud. The NPTC was used to design a convolutional neural network (NPTC-net) for point cloud classification and segmentation. Comparisons with state-of-the-art methods indicated that the proposed NPTC-net is competitive with the best existing methods.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Adalsteinsson, D., Sethian, J.A.: A fast level set method for propagating interfaces. J. Comput. Phys. **118**(2), 269–277 (1995)
3. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1534–1543. IEEE Computer Society, Los Alamitos, CA, USA (2016).10.1109/CVPR.2016.170. https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.170
4. Bindu, V., Nair, K.R.: A fast narrow band level set formulation for shape extraction. In: The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014), pp. 137–142. IEEE (2014)
5. Boscaini, D., Masci, J., Rodolà, E., Bronstein, M.: Learning shape correspondence with anisotropic convolutional neural networks. In: D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (eds.) Advances in Neural Information Processing Systems 29, pp. 3189–3197. Curran Asso-

ciates, Inc. (2016). http://papers.nips.cc/paper/6045-learning-shape-correspondence-with-anisotropic-convolutional-neural-networks.pdf
6. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017). https://doi.org/10.1109/MSP.2017.2693418
7. Bruna, J., Zaremba, W., Szlam, A., Lecun, Y.: Spectral networks and locally connected networks on graphs. In: International Conference on Learning Representations (ICLR2014), CBLS, April 2014 (2014)
8. Cao, W., Yan, Z., He, Z., He, Z.: A comprehensive survey on geometric deep learning. IEEE Access **8**, 35929–35949 (2020)
9. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (eds.) Advances in Neural Information Processing Systems 29, pp. 3844–3852. Curran Associates, Inc. (2016). http://papers.nips.cc/paper/6081-convolutional-neural-networks-on-graphs-with-fast-localized-spectral-filtering.pdf
10. Dong, B.: Sparse representation on graphs by tight wavelet frames and applications. Appl. Comput. Harmonic Anal. **42**(3), 452–479 (2017)
11. Eldar, Y., Lindenbaum, M., Porat, M., Zeevi, Y.Y.: The farthest point strategy for progressive image sampling. IEEE Trans. on Image Process. **6**(9), 1305–1315 (1997)
12. Franke, R., Nielson, G.M.: Scattered data interpolation and applications: a tutorial and survey. Geomet. Model. pp. 131–160 (1991)
13. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
14. Grevera, G.J., Udupa, J.K.: An objective comparison of 3-d image interpolation methods. IEEE Trans. Med. Imaging **17**(4), 642–652 (1998)
15. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Appl. Comput. Harmonic Anal. **30**(2), 129–150 (2011)
16. Helgason, S.: Differential geometry, Lie groups, and symmetric spaces, vol. 80. Academic press (1979)
17. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163 (2015)
18. Hojjatoleslami, S., Kittler, J.: Region growing: a new approach. IEEE Trans. Image process. **7**(7), 1079–1084 (1998)
19. Jiang, M., Zhong, Y., Wang, X., Huang, X., Guo, R.: Improve the nonparametric image segmentation with narrowband levelset and fast gauss transformation (2012)
20. Katznelson, Y.: An introduction to harmonic analysis. Cambridge University Press (2004)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). arXiv:1412.6980
22. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 863–872. IEEE Computer Society, Los Alamitos, CA, USA (2017). 10.1109/ICCV.2017.99. https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.99
23. Knebelman, M.: Spaces of relative parallelism. Ann. Math. pp. 387–399 (1951)
24. Kobayashi, S., Nomizu, K.: Foundations of differential geometry, vol. 2. Interscience publishers New York (1969)
25. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
26. Lai, R., Liang, J., Zhao, H.: A local mesh method for solving pdes on point clouds. Inverse Prob. Imaging **7**(3), 737–755 (2013)
27. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature 521(7553), 436 (2015)
28. Levie, R., Monti, F., Bresson, X., Bronstein, M.M.: Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Trans. Sig. Process. **67**(1), 97–109 (2019). https://doi.org/10.1109/TSP.2018.2879624
29. Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
30. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (eds.) Advances in Neural Information Processing Systems 31, pp. 820–830. Curran Associates, Inc. (2018). http://papers.nips.cc/paper/7362-pointcnn-convolution-on-x-transformed-points.pdf
31. Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M.A., Cao, D., Li, J.: Deep learning for lidar point clouds in autonomous driving: a review. IEEE Transactions on Neural Networks and Learning Systems (2020)

32. Liu, W., Sun, J., Li, W., Hu, T., Wang, P.: Deep learning on point clouds and its application: A survey. Sensors **19**(19), 4188 (2019)
33. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
34. Mao, J., Wang, X., Li, H.: Interpolated convolutional networks for 3d point cloud understanding. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1578–1587 (2019)
35. Masci, J., Boscaini, D., Bronstein, M.M., Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds. In: The IEEE International Conference on Computer Vision (ICCV) Workshops (2015)
36. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
37. Morgan, H.L.: The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. J. Chem. Document. **5**(2), 107–113 (1965)
38. Osher, S., Fedkiw, R.: Level set methods and dynamic implicit surfaces, vol. 153. Springer Science & Business Media (2006)
39. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. J. Comput. Phys. **79**(1), 12–49 (1988)
40. Pal, S.K., Mitra, S.: Multilayer perceptron, fuzzy sets, classifiaction (1992)
41. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
42. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30, pp. 5099–5108. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature-learning-on-point-sets-in-a-metric-space.pdf
43. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
44. Rosenthal, P., Linsen, L.: Smooth surface extraction from unstructured point-based volume data using pdes. IEEE Trans. Visual. Comput. Graph. **14**(6), 1531–1546 (2008)
45. Rosenthal, P., Molchanov, V., Linsen, L.: A narrow band level set method for surface extraction from unstructured point-based volume data
46. Schonsheck, S.C., Dong, B., Lai, R.: Parallel transport convolution: A new tool for convolutional neural networks on manifolds. CoRR **abs/1805.07857** (2018). arXiv:1805.07857
47. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proceed. Natl. Acad. Sci. **93**(4), 1591–1595 (1996)
48. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. J. Big Data **6**(1), 1–48 (2019)
49. Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.Y.: Tangent convolutions for dense prediction in 3d. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
50. Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S.: Segcloud: Semantic segmentation of 3d point clouds. In: 2017 International Conference on 3D Vision (3DV), pp. 537–547 (2017). 10.1109/3DV.2017.00067
51. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 6411–6420 (2019)
52. Wang, C., Samari, B., Siddiqi, K.: Local spectral graph convolution for point set feature learning. In: The European Conference on Computer Vision (ECCV) (2018)
53. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2589–2597 (2018)
54. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. CoRR **abs/1801.07829** (2018). arXiv:1801.07829
55. Wu, W., Qi, Z., Li, F.: Pointconv: Deep convolutional networks on 3d point clouds. CoRR **abs/1811.07246** (2018). arXiv:1811.07246
56. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
57. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: The European Conference on Computer Vision (ECCV) (2018)

58. Yang, Y., Liu, S., Pan, H., Liu, Y., Tong, X.: Pfcnn: convolutional neural networks on 3d surfaces using parallel frames. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13578–13587 (2020)
59. Yang, Z., Litany, O., Birdal, T., Sridhar, S., Guibas, L.: Continuous geodesic convolutions for learning on 3d shapes. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 134–144 (2021)
60. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Trans. Graph. **35**(6), 210:1–210:12 (2016). 10.1145/2980179.2980238. http://doi.acm.org/10.1145/2980179.2980238
61. Zhang, C., Luo, W., Urtasun, R.: Efficient convolutions for real-time semantic segmentation of 3d point clouds. In: 2018 International Conference on 3D Vision (3DV), pp. 399–408. IEEE (2018)
62. Zhao, H.: A fast sweeping method for eikonal equations. Math. Comput. **74**(250), 603–627 (2005)

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com