DiffusionNet: Discretization Agnostic Learning on Surfaces

NICHOLAS SHARP, Carnegie Mellon University, University of Toronto
SOUHAIB ATTAIKI, LIX, École Polytechnique
KEENAN CRANE, Carnegie Mellon University
MAKS OVSJANIKOV, LIX, École Polytechnique

We introduce a new general-purpose approach to deep learning on 3D surfaces, based on the insight that a simple diffusion layer is highly effective for spatial communication. The resulting networks are automatically robust to changes in resolution and sampling of a surface—a basic property which is crucial for practical applications. Our networks can be discretized on various geometric representations such as triangle meshes or point clouds, and can even be trained on one representation then applied to another. We optimize the spatial support of diffusion as a continuous network parameter ranging from purely local to totally global, removing the burden of manually choosing neighborhood sizes. The only other ingredients in the method are a multi-layer perceptron applied independently at each point, and spatial gradient features to support directional filters. The resulting networks are simple, robust, and efficient. Here, we focus primarily on triangle mesh surfaces, and demonstrate state-of-the-art results for a variety of tasks including surface classification, segmentation, and non-rigid correspondence.

CCS Concepts: • **Computing methodologies** → **Shape analysis**.

Additional Key Words and Phrases: geometric deep learning, geometry processing, discrete differential geometry, partial differential equations

ACM Reference Format:
Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2022. DiffusionNet: Discretization Agnostic Learning on Surfaces. *ACM Trans. Graph.* 1, 1, Article 1 (January 2022), 16 pages. https://doi.org/10.1145/3507905

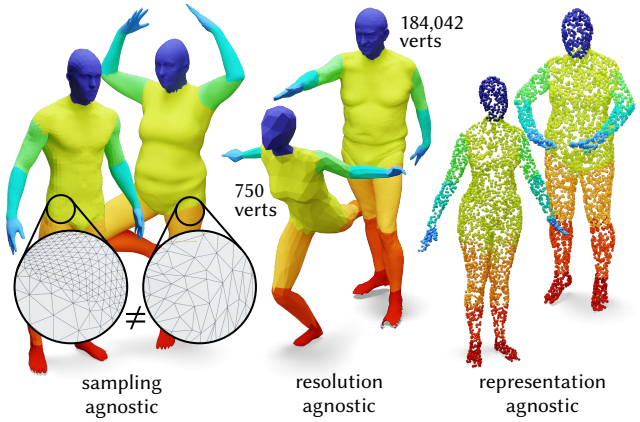


Fig. 1. Surface learning methods must generalize to shapes represented differently from the training set to be useful in practice, yet many existing approaches depend strongly on mesh connectivity. Here, our DiffusionNet trained for human segmentation with limited variability seen during training automatically generalizes to widely varying mesh samplings (*left*), scales gracefully to resolutions ranging from a simplified model to a large raw scan (*middle*), and can even be evaluated directly on point clouds (*right*).

1 INTRODUCTION

Recently there has been significant interest in learning techniques for non-uniform geometric data, inspired by the tremendous success of convolutional neural networks (CNNs) in computer vision. A particularly challenging setting is extending the power of CNNs to learning directly on curved surfaces [Bronstein et al. 2017; Hanocka et al. 2019; Masci et al. 2015; Poulenard and Ovsjanikov 2018]. Unlike volumetric [Maturana and Scherer 2015] or point-based [Qi et al. 2017a] approaches, surface-based methods exploit the connectivity of the surface representation to improve performance, and furthermore can be robust in the presence of non-rigid deformations, making them a strong solution for many tasks such as deformable shape matching [Boscaini et al. 2016; Masci et al. 2015].

However, although the field has largely been focused on the benchmark accuracy of networks for such problems, at least two other major roadblocks remain for achieving the full potential of learning on surfaces. First, whereas real-world geometric data comes from a variety of sources, existing networks are strongly tied to a particular representation (*e.g.*, triangulations *or* point clouds) or even discretization resolution. Hence, training cannot benefit from all available data. One popular strategy is to simply convert all data to a common representation (*e.g.*, via point sampling), but this approach has well-known drawbacks (sampling a high-quality, detailed surface can alias thin features, lose informative details, *etc.*). Second, many existing mesh-based architectures do not scale well to high-resolution surface data. Though coarse inputs are sufficient for, *e.g.*, classification tasks, they preclude potential future applications such as high-fidelity geometric analysis and synthesis.

A key technical difficulty in surface-based learning is defining appropriate notions of convolution and pooling—two main building blocks in traditional CNNs. Unfortunately, unlike the Euclidean case, there is no universal canonical notion of convolution on surfaces. Existing approaches have tried to address this challenge through a variety of solutions such as mapping to a canonical domain [Maron et al. 2017; Sinha et al. 2016], exploiting local parametrizations [Boscaini et al. 2016; Masci et al. 2015; Wiersma et al. 2020] or applying convolution on the edges of the mesh [Hanocka et al. 2019]. However, the

Authors' addresses: Nicholas Sharp, Carnegie Mellon University, University of Toronto; Souhaib Attaiki, LIX, École Polytechnique; Keenan Crane, Carnegie Mellon University; Maks Ovsjanikov, LIX, École Polytechnique .

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2022 Copyright held by the owner/author(s).
0730-0301/2022/1-ART1
https://doi.org/10.1145/3507905

use of more advanced and delicate geometric operations, such as computing geodesics or parallel transport, has a significant impact on both the robustness and scalability of the resulting methods. Perhaps even more importantly, existing surface-based approaches are often *too* sensitive to the underlying mesh structure, and thus unable to generalize to significantly different sampling and triangulations between training and test sets. As a result, despite significant recent progress in geometric deep learning [Cao et al. 2020; Greengard 2020], current methods typically struggle to cope with the variability, complexity and scale encountered in the real-world surface and mesh-based settings.

In this work, we propose a method that exploits the surface representation, but is both scalable and robust in the presence of significant sampling changes (see Figure 1). Our main observation is that expensive and potentially brittle operations used in previous works [Masci et al. 2015; Poulenard and Ovsjanikov 2018; Wiersma et al. 2020] can be replaced with two basic geometric operations: a learned diffusion layer for information propagation and a spatial gradient for capturing anisotropy. Discretizing these operations with the principled techniques of discrete differential geometry [Crane et al. 2013; Meyer et al. 2003] then *automatically* endows the resulting networks with both robustness and scalability, while maintaining the simplicity of the learning framework.

Remarkably, we show that combining these basic geometric operations yields neural networks that are not only robust and scalable, but also achieve state-of-the-art results in a wide variety of applications, including deformable surface segmentation, classification, as well as unsupervised and supervised non-rigid shape matching. Perhaps even more fundamentally, our DiffusionNet offers a unified perspective across representations of surface geometry—in principle it can be applied to any geometric representation where one has a Laplacian and gradient operator. In this paper, for example, we show how the same architecture achieves accurate results for both meshes and point clouds, and even allows training on one and evaluating on the other.

Contributions. The main contributions of this work are:

- We show that a simple learned diffusion operation is sufficient to share spatial data in surface learning.
- We introduce spatial gradient features for learning local directional filters.
- Inspired by these insights, we present DiffusionNet, an architecture for learning on surfaces which has many advantages including robustness to discretization, and achieves state-of-the-art results on several benchmarks.

2 RELATED WORK

Applying deep learning techniques to 3D shapes is a rich and extensive area of research. Below we review the approaches most closely related to ours, and refer the interested readers to recent surveys, including [Bronstein et al. 2017; Cao et al. 2020; Xu et al. 2016].

View-based and volumetric methods. Most early geometric deep learning-based methods directly leveraged tools developed for 2D images, and thus mapped 3D shapes onto the plane either using multi-view renderings [Kalogerakis et al. 2017; Su et al. 2015; Wei et al. 2016] or more global, often parametrization-based techniques such as panoramas [Sfikas et al. 2017; Shi et al. 2015], geometry images [Sinha et al. 2016], or metric-preserving mappings [Ezuz et al. 2017], among many others.

Another direct approach to applying convolution to 3D shapes relies on volumetric voxel grid representations, which has led to a variety of methods, including [Maturana and Scherer 2015; Wu et al. 2015] and their efficient extensions [Klokov and Lempitsky 2017; Wang et al. 2017]. Such techniques can, however, be computationally expensive and difficult to apply to detailed deformable shapes.

2.1 Learning on Surfaces

Methods that learn on 3D surfaces directly typically fall into two major categories, based either on point cloud or triangle mesh representations.

Point-based methods. A successful set of methods for learning on 3D shapes represented as point clouds was pioneered by the PointNet [Qi et al. 2017a] and PointNet++ [Qi et al. 2017b] architectures, which have been extended in many recent works, including PointCNN [Li et al. 2018], DGCNN [Wang et al. 2019], PCNN [Atzmon et al. 2018] and KPConv [Thomas et al. 2019] to name a few (see also [Guo et al. 2020] for a recent survey). Moreover, recent efforts have also been made to incorporate invariance and equivariance of the networks with respect to various geometric transformations, *e.g.*, [Deng et al. 2018; Hansen et al. 2018; Li et al. 2021; Poulenard et al. 2019; Zhang et al. 2019; Zhao et al. 2020]. The major advantages of point-based methods are their simplicity, flexibility, applicability in a wide range of settings and robustness in the presence of noise and outliers. However, first, their overall accuracy can often be lower than that of methods that explicitly use surface (*e.g.*, mesh) connectivity when it is available. Second, though effective on static mechanical objects and scenes, point-based methods may not be well-suited for *deformable* (non-rigid) shape analysis, requiring extremely large training sets and significant data augmentation to achieve good results, *e.g.*, for non-rigid shape matching applications [Donati et al. 2020; Groueix et al. 2018]. Globally supported point-based networks were recently considered in [Peng et al. 2020]; our method naturally allows global support via learned diffusion.

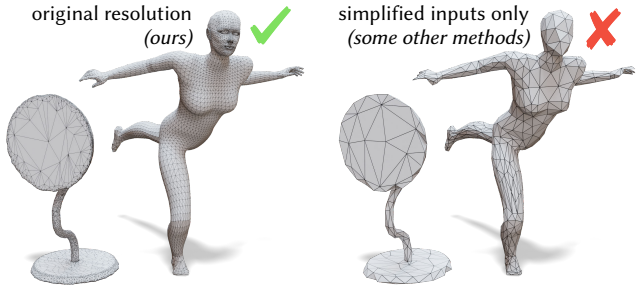


Fig. 2. Many recent mesh-based learning methods are applied only to dramatically simplified inputs (Section 5.6), while our method easily processes full-resolution models, preserving detail and facilitating adoption.

Surface and graph-based methods. To address the limitations of point-based approaches, several methods have been proposed that operate directly on mesh surfaces and thus can learn filters that are intrinsic and robust to complex non-rigid deformations. The earliest pioneering approaches in this direction generalize convolutions [Boscaini et al. 2016; Fey et al. 2018; Masci et al. 2015; Monti et al. 2017], typically using local surface parameterization via the logarithmic map. Unfortunately local parameterizations are only defined up to rotation in the tangent plane, leading to several methods which address this issue through design of *equivariant* surface networks [Haan et al. 2021; He et al. 2020; Mitchel et al. 2021; Poulenard and Ovsjanikov 2018; Wiersma et al. 2020; Yang et al. 2021]. Likewise, operating on vector-valued data in a local tangent space can expand the expressivity of the filter space [Beani et al. 2021; Mitchel et al. 2021; Wiersma et al. 2020]. Our method leverages learned gradient features (Section 3.4), which geometrically require only a local spatial gradient operation, and sidestep the challenge of equivariant filters by using only inner products, which are naturally invariant. These gradient features are built on local differential operators, which have also been exploited in other recent methods (*e.g.*, [Eliasof and Treister 2020; Jiang et al. 2018]).

Surface mesh structure has also been used in variety of graph-like approaches which specifically leverage mesh connectivity [Bodnar et al. 2021; Feng et al. 2019; Gong et al. 2019; Hajij et al. 2020; Hanocka et al. 2019; Lim et al. 2018; Milano et al. 2020; Verma et al. 2018], the structure of discrete operators [Smirnov and Solomon 2021], or even random walks along edges [Lahav and Tal 2020], among others. While accurate, these methods can be costly on densely sampled shapes and often are not robust to significant changes in the mesh structure (Figure 3).

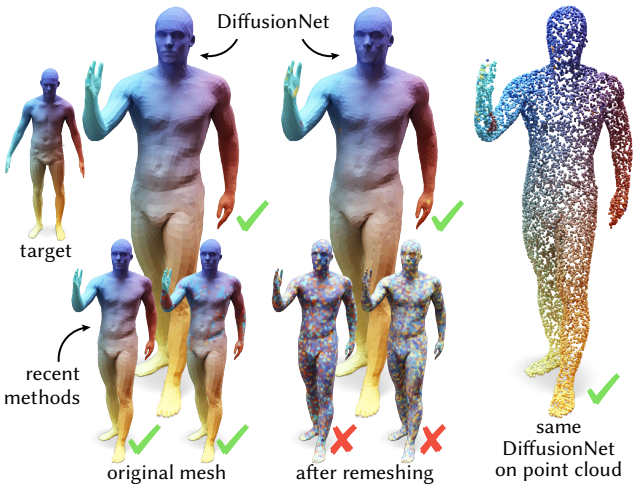


Fig. 3. Although past methods have achieved high-accuracy benchmark results for learning on meshes [Fey et al. 2018; Li et al. 2020b], they are prone to over-fitting to the mesh connectivity, rather than learning the underlying shape structure (Section 5.4). In contrast, DiffusionNet learns an accurate representation-agnostic solution, which even supports training on meshes and evaluating on a point cloud (*last column*).

2.2 Spectral Methods

Our use of diffusion is also loosely related to techniques that operate in the spectral domain and often exploit the link between convolution and operations in a derived (*e.g.*, Fourier or Laplace-Beltrami) basis, including [Bruna et al. 2014; Levie et al. 2018; Sun et al. 2020]. Such methods have a long history in graph-based learning and are well-rooted in data analysis more broadly, including Laplacian eigenmaps [Belkin and Niyogi 2003], spectral clustering [Vallet and Lévy 2008] and diffusion maps [Coifman et al. 2005]. In geometry processing, spectral methods have been used for a range of tasks including multi-resolution representation [Levy 2006], segmentation [Rustamov 2007] and matching [Ovsjanikov et al. 2012], among others [Vallet and Lévy 2008; Zhang et al. 2010].

Unfortunately, Laplacian eigenfunctions depend on each shape and thus coefficients or learned filters from one shape are not trivially transferable to another. Levie et al. [2019] argue for transfer between discretizations of the same shape, but 3D geometric learning typically demands transfer between different shapes. Functional maps [Ovsjanikov et al. 2012] can be used to "translate" coefficients between shapes, and have been used, *e.g.*, in [Yi et al. 2017] with spectral filter learning. Deep functional maps [Litany et al. 2017] propose to learn features in the primal domain, which are then projected onto the Laplace-Beltrami basis for functional map estimation. However, the features are still learned either with MLPs starting from pre-computed descriptors [Ginzburg and Raviv 2020; Halimi et al. 2019; Litany et al. 2017; Roufosse et al. 2019] or using point-based architectures [Donati et al. 2020].

Instead, we propose an approach that learns the parameters of a diffusion process which is directly transferable across shapes and, as we show below, can be used effectively in applications like non-rigid shape matching. We also stress that DiffusionNet *is not* spectral in nature and only uses spectral operations as an acceleration technique for evaluating diffusion efficiently.

We also note that our use of the Laplacian in defining the diffusion operator is related to methods based on polynomials of the Laplacian [Defferrard et al. 2016; Kostrikov et al. 2018], CayleyNets [Levie et al. 2018] and their recent application in shape matching using ACSCNNs [Li et al. 2020b]. However, we demonstrate that complex polynomial filters can be replaced with simple learned diffusion, and moreover that gradient features can inject orientation information into the network, improving performance and robustness.

Similarly to our approach, diffusion for smooth communication has been explored on graphs [Klicpera et al. 2019; Xu et al. 2019], images [Liu et al. 2016], and point clouds [Hansen et al. 2018]. In contrast, our method directly learns a diffusion time per-feature (which significantly improves performance, Table 7), incorporates a learned gradient operation, and is applied directly to mesh surfaces.

Pooling. In surface learning it is nontrivial to define pooling, especially on meshes where it often amounts to mesh simplification [Hanocka et al. 2019]. Various recent operations have been proposed for point cloud [Hu et al. 2020; Lin et al. 2020], mesh [Milano et al. 2020; Zhou et al. 2020] or even graph pooling [Li et al. 2020a; Ma et al. 2020]. A key advantage of our approach is that it automatically supports global spatial support without any downsampling operation, simplifying implementation and improving learning.

3 METHOD

Our method consists of three main building blocks: multi-layer perceptrons (MLPs) applied at each point to model pointwise scalar functions of feature channels, a learned diffusion operation for propagating information across the domain, and local spatial gradient features to expand the network's filter space beyond radiallly-symmetric filters. In this section, we describe these main numerical components, and then we assemble them into an effective architecture in Section 4. Our method is defined in a representation-agnostic manner; applying it to meshes or point clouds simply amounts to assembling the appropriate Laplacian and gradient matrices as we discuss below.

3.1 Pointwise Perceptrons

On a mesh or point cloud with V vertices, we consider a collection of D scalar features defined at each vertex. Our first basic building block is a pointwise function $f : \mathbb{R}^D \to \mathbb{R}^D$, which is applied independently at every vertex to transform the features. We represent these pointwise functions as a standard multilayer perceptron (MLP) with shared weights across all vertices. Although these MLPs can fit arbitrary functions at each point, they do not capture the spatial structure of the surface, or allow any communication between vertices, so a richer structure is needed.

Past approaches for communication have ranged from global reductions to explicit geodesic convolutions—instead, we will demonstrate that a simple learned diffusion layer effectively propagates information, without the need for potentially costly or error-prone computations.

3.2 Learned Diffusion

In the continuous setting, diffusion of a scalar field u on a domain is modeled by the *heat equation*

$$\frac{d}{dt}u_t = \Delta u_t, \tag{1}$$

where Δ is the Laplacian (or more formally: the *Laplace-Beltrami operator*). The action of diffusion can be represented via the heat operator H_t, which is applied to some initial distribution u_0 and produces the diffused distribution u_t; this action can be defined as $H_t(u_0) = \exp(t\Delta)u_0$, where exp is the operator exponential. Over time, diffusion is an increasingly-global smoothing process: for $t = 0$, H_t is the identity map, and as $t \to \infty$ it approaches the average over the domain.

We propose to use the heat equation to spatially propagate features for learning on surfaces; its principled foundations ensure that results are largely invariant to the way the surface is sampled or meshed. To discretize diffusion, one replaces Δ with the weak Laplace matrix L and mass matrix M. Here, L is a positive semi-definite sparse matrix $L \in \mathbb{R}^{V\times V}$ with the opposite sign convention such that $M^{-1}L \approx -\Delta$. The number of entries in L and M are generally $O(V)$, scaling effectively to large inputs (Table 6). On triangle meshes, we will use the *cotan-Laplace* matrix, which is ubiquitous in geometry processing applications [Crane et al. 2013; MacNeal 1949; Pinkall and Polthier 1993]; for point clouds we will use the related Laplacian from [Sharp and Crane 2020]. This matrix has also been defined for voxel grids [Caissard et al. 2019], polygon meshes [Bunge et al. 2020], tetrahedral meshes [Alexa et al. 2020], *etc.* The weak Laplace matrix is accompanied by a mass matrix M, such that the rate of diffusion is given by $-M^{-1}Lu$. Here M will be a "lumped" diagonal matrix of areas associated with each vertex.

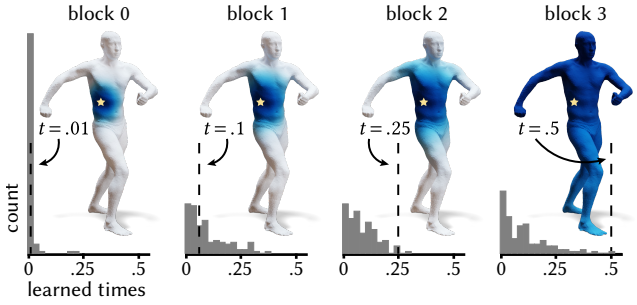


Fig. 4. We propose to learn a diffusion time for each feature channel, automatically tuning spatial support during training. The histograms show the learned times at each block in a DiffusionNet trained for segmentation; the times marked by the dashed lines are visualized by diffusing a point source from the starred point. The first block uses mainly local diffusion, while a channel in the last block finds nearly global support.

We define a learned diffusion layer $h_t : \mathbb{R}^V \to \mathbb{R}^V$, which diffuses a feature channel u for learned time $t \in \mathbb{R}_{\geq 0}$. In our networks, $h_t(u)$ is applied independently to each feature channel, with a separate learned time t per-channel. Learning the diffusion parameter is a key strength of our method, allowing the network to continuously optimize for spatial support ranging from purely local to totally global, and even choose different receptive fields for each feature (Figure 4). We thus sidestep challenges like manually choosing the support radius of a convolution, or sizes for a pooling hierarchy.

In the language of deep learning, diffusion can be viewed as a kind of smooth mean (average) pooling operation with many benefits: it has a geometrically-principled meaning, its support ranges from purely local to totally global via the choice of diffusion time, and it is differentiable with respect to diffusion time, allowing spatial support to be automatically optimized as a network parameter.

A note on generality. Remarkably, eschewing traditional representations of convolutions in favor of diffusion does not reduce the expressive power of our networks. This is supported by the following theoretical result (that we prove in Appendix A), which shows that radially-symmetric convolutions are contained in the function space defined by diffusion followed by a pointwise map:

Lemma 1 (Inclusion of radially-symmetric convolutions). *For a signal $u : \mathbb{R}^2 \to \mathbb{R}$, let $U_r(p) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denote the integral of u along the r-sphere at p, and let $u_t(p) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denote the signal value at p after diffusion for time t. Then there exists a function transform $\mathcal{T}$ which recovers $U_r(p)$ from $u_t(p)$*

$$U_r(p) = \mathcal{T}[u_t(p)](r).$$

Thus convolution with a radial kernel $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is given by

$$(u * \alpha)(p) = \int_{\mathbb{R}^2} \alpha(|q - p|)u_0(q)\, dq = \int_0^\infty \alpha(r)\mathcal{T}[u_t(p)](r)\, dr,$$

which is a pointwise operation at p on the diffused values $u_t(p)$.

This fact is significant because it suggests that simple and robust diffusion can, without loss of generality, be used to replace complicated operations such as radial geodesic convolution.

Importantly, we will also extend our architecture beyond radially-symmetric filters by incorporating gradient features (Section 3.4).

3.3 Computing Diffusion

Many numerical schemes could potentially be used to evaluate the diffusion layer $h_t(u)$, from direct solvers [Chen et al. 2008] to hierarchical schemes [Liu et al. 2021; Vaxman et al. 2010]. In particular, we seek schemes which are efficient as well as differentiable, to enable network training. Here we describe two simple methods considered in our experiments. The first scheme we consider is an implicit timestep, which is straightforward but requires solving large sparse linear systems, and the second is spectral expansion, which uses only efficient dense arithmetic at evaluation time but requires some modest precomputation. Both are easily implemented using common numerical libraries, and we observe that networks trained with either approach have similar accuracy. Efficiency is evaluated in Section 5.6; we generally recommend spectral acceleration.

3.3.1 Direct Implicit Timestep. Perhaps the simplest effective approach to simulate diffusion is a single implicit Euler timestep

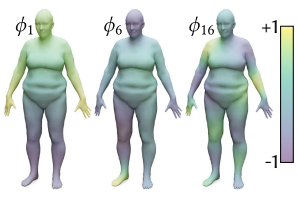
$$h_t(u) := (M + tL)^{-1} Mu \tag{2}$$

which amounts to solving a (sparse) linear system for each diffusion operation. Using an implicit backward timestep rather than an explicit forward timestep is crucial as it makes the scheme stable, allows global support, and yields a reasonable approximation of diffusion after just one step. Solving linear systems (including derivative computation) is supported in modern learning software frameworks, allowing Equation 2 to implement a learnable diffusion block. However, this amounts to solving a distinct large linear system for each channel, and GPU-based computation may fall back to solving dense linear systems, which means that direct implicit timesteps may not scale well to large problems.

3.3.2 Spectral Acceleration. An alternate approach is to leverage a closed-form expression for diffusion in the basis of low-frequency Laplacian eigenfunctions [Vallet and Lévy 2008; Zhang et al. 2010]. Once the eigenbasis has been precomputed, diffusion can then be evaluated for any time t via elementwise exponentiation. Truncating diffusion to a low-frequency basis incurs some approximation error, but we find that this approximation has little effect on our method (Figure 13), perhaps because diffusion quickly damps high-frequency content regardless.

For weak Laplace matrix L and mass matrix M, the eigenvectors $\phi_i \in \mathbb{R}^V$ are solutions to:

$$L\phi_i = \lambda_i M \phi_i, \tag{3}$$



corresponding to the first k smallest-magnitude eigenvalues $\lambda_1, \ldots, \lambda_k$. We normalize them so that $\phi_i^T M \phi_i = 1$. These eigenvectors are easily precomputed for each shape of interest via standard numerical packages [Lehoucq et al. 1998]; the inset figure shows several example functions ϕ_i for a surface of a human shape.

Let $\Phi := [\phi_i] \in \mathbb{R}^{V \times k}$ be the stacked matrix of eigenvectors, which form an orthonormal basis with respect to M. We can then project any scalar function u to obtain its coefficients c in the spectral basis via $c \leftarrow \Phi^T M u$, and recover values at vertices as $u \leftarrow \Phi c$. Conveniently, diffusion for time t is easily expressed as an elementwise scaling of spectral coefficients according to $c_i \leftarrow e^{-\lambda_i t} c_i$. The diffusion layer $h_t(u)$ is then evaluated by projecting on to the spectral basis, evaluating pointwise diffusion, and projecting back

$$h_t(u) := \Phi \begin{bmatrix} e^{-\lambda_0 t} \\ e^{-\lambda_1 t} \\ \ldots \end{bmatrix} \odot (\Phi^T M u) \tag{4}$$

where $\odot$ denotes the Hadamard (elementwise) product. This operation is efficiently evaluated using dense linear algebra operations like elementwise exponentiation and matrix multiplication, and is easily differentiable with respect to u and t.

Remarks. We emphasize that DiffusionNet can still learn high-frequency outputs despite the use of a low-frequency basis (*e.g.*, in Section 5.4). Intuitively, diffusion is used for communication across points, for which a low-frequency approximation is typically sufficient, while MLPs and gradient features learn high-frequency features as needed for a task. Additionally, we note that DiffusionNet is *not* a spectral learning method—spectral coefficients are never used to represent filters or latent data, and thus no issues arise due to differing eigenbases on different shapes. Spectral acceleration is merely one possible numerical scheme to compute diffusion.

3.4 Spatial Gradient Features

Our learned diffusion layer enables propagation of information across different points on a shape, but it supports only radially-symmetric filters about a point. The last building block in our method enables a larger space of filters by computing additional features from the spatial gradients of signal values at vertices (Figure 6). Specifically, we construct features from the inner products between pairs of feature gradients at each vertex, after applying a learned scaling or rotation.

Evaluating gradients. We will express the spatial gradient of a scalar function on a surface as a 2D vector in the tangent space of each vertex. These gradients can be evaluated by a standard procedure, choosing a normal vector at each vertex (given as input or locally approximated), then projecting neighbors into the tangent plane—either 1-ring neighbors on a mesh, or m-nearest neighbors in a point cloud. The gradient is then computed in the tangent plane via least-squares approximation of the function values at neighboring points (see Mukherjee and Wu [2006] for analysis). These gradient operators at each vertex can be assembled into a sparse matrix $G \in \mathbb{C}^{V \times V}$, which is applied to a vector u of real values at vertices to produce gradient tangent vectors at each vertex. This matrix does not depend on the features, and can be precomputed once for each shape. We use complex numbers as a convenient notation for tangent vectors, in an arbitrary reference basis in the tangent plane of each point (as in [Knöppel et al. 2013; Sharp et al. 2019], *etc.*). If the normals are consistently oriented, then the imaginary axis is chosen to form a right-handed basis in 3D with respect to the outward surface normal.

Learned pairwise products. Equipped with per-vertex spatial gradients of each channel, we learn informative scalar features by evaluating an inner product between pairs of feature gradients at each vertex, after a learned linear transformation. Inner products are invariant to rotations of the coordinate system, so these features are invariant to the choice of tangent basis at vertices, as expected. Putting it all together, given a collection of D scalar feature channels, for each channel u we first construct its spatial gradient as $z_u \in \mathbb{C}^V$, a vector of local 2D gradients per-vertex

$$z_u := Gu \tag{5}$$

then at each vertex v, we stack the local gradients of all channels to form $w_v \in \mathbb{C}^D$ and obtain real-valued features $g_v \in \mathbb{R}^D$ as

$$g_v := \tanh(\mathrm{Re}(\overline{w}_v \odot A w_v)) \tag{6}$$

where A is a learned square $D \times D$ matrix, and taking the real part Re after a complex conjugate $\overline{w}_v$ is again just a notational convenience for dot products between pairs of 2D vectors. This means the i^{th} entry of the output at vertex v is given by the dot product $g_v(i) = \tanh\left(\mathrm{Re}\left\{\sum_{j=1}^{D} \overline{w}_v(i) A_{ij} w_v(j)\right\}\right)$, so that each inner product is scaled by a learned coefficient A_{ij}. The outer $\tanh(\cdot)$ nonlinearity is not fundamental, but we find that it stabilizes training.

The choice of A as a complex or real matrix has a subtle relationship with the orientation of the underlying surface. Multiplying $w_v(j)$ by a *complex* scalar both rotates and scales local gradient vectors before taking inner products (recall that complex multiplication can be interpreted as a rotation and scaling in the complex plane). In contrast, real A only allows scaling. However, the direction of rotation (clockwise or counter-clockwise) depends on the choice of the outward normal, and hence on orientation—so surfaces with consistently oriented normals gain a richer representation by learning a complex matrix, whereas surfaces without consistent orientation (*e.g.*, raw point clouds) should restrict to real A.

In Figure 5, a small synthetic experiment (detailed in Appendix C) demonstrates how these learned rotations allow our method to disambiguate bilateral symmetry even in a purely intrinsic representation, a common challenge in non-rigid shape correspondence.

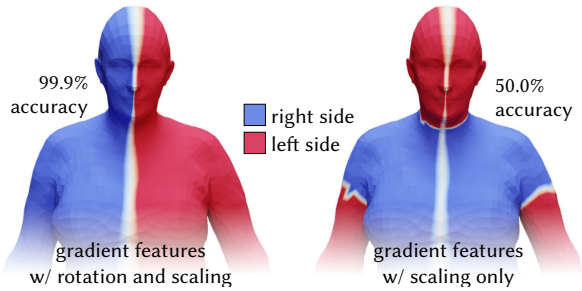


Fig. 5. In a synthetic experiment, we demonstrate how our networks can successfully segment the left and right sides of bilaterally symmetric models even in a purely-intrinsic formulation (*left*), because rotation in the tangent space for gradient features encodes a notion of orientation. Replacing this rotation with scaling (i.e., using a real matrix A in Equation 6) removes the sensitivity to shape orientation (*right*), but also avoids the need for consistent outward normals. See Appendix C for details.

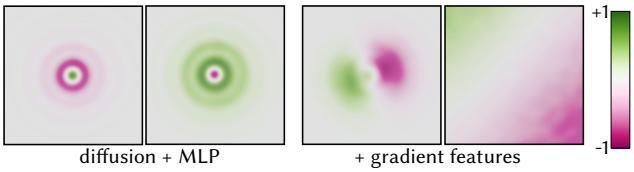


Fig. 6. Diffusion followed by an MLP enables the network to learn radially-symmetric filters (*left*); introducing gradient features expands the space to include directional filters (*right*), while remaining invariant to the choice of local tangent basis. Here, we take a DiffusionNet block trained for segmentation and visualize the learned filter via channels of a normalized signal which maximizes the block output at the center point.

4 DIFFUSIONNET ARCHITECTURE

The previous section establishes three ingredients for learning on surfaces: an MLP applied independently at each vertex to represent pointwise functions, learned diffusion for spatial communication, and spatial gradient features to model directional filters. We combine these ingredients to construct *DiffusionNet* (Figure 7), composed of several *DiffusionNet blocks*. This simple network operates on a fixed channel width D of scalar values throughout, with each DiffusionNet block diffusing the features, constructing spatial gradient features, and feeding the result to an MLP.

We include residual connections to stabilize training [He et al. 2016], as well as linear layers to convert to the expected input and output dimension. When appropriate, results at the edges or faces of a mesh can be computed by averaging network outputs from the incident vertices, *e.g.*, to segment the faces of a mesh. Various activations can be appended to the end of the network based on the problem at hand, such as a softmax for segmentation, or a global mean followed by a softmax for classification; otherwise, this same architecture is used for all experiments.

Remarkably, we do not find it necessary to use any spatial convolutions or pooling hierarchies on surfaces—avoiding these potentially-complex operations helps keep DiffusionNet simple and robust.

Invariance. DiffusionNet is invariant to rigid motion of the underlying shapes as long as the input features remain unchanged, due to the intrinsic geometric nature of diffusion and spatial gradients. The overall invariance then depends on the choice of input features.

4.1 Input features

DiffusionNet takes a vector of scalar values per-vertex as input features. Here we consider two simple choices of features, others could easily be included when available. Most directly, we simply use the raw 3D coordinates of a shape as input; rotation augmentation can be used to promote rigid invariance when inputs are not consistently aligned. When rigid or even non-rigid invariance is desired, we instead use the Heat Kernel Signatures [Sun et al. 2009] as input; these signatures are trivially computed from the spectral basis in Section 3.3.2. Due to the intrinsic nature of our approach, with HKS as input, the networks are invariant to any orientation-preserving isometric deformation of the shape. Higher-order descriptors such as SHOT [Tombari et al. 2010] seem unnecessary, and may be unstable under remeshing [Donati et al. 2020].

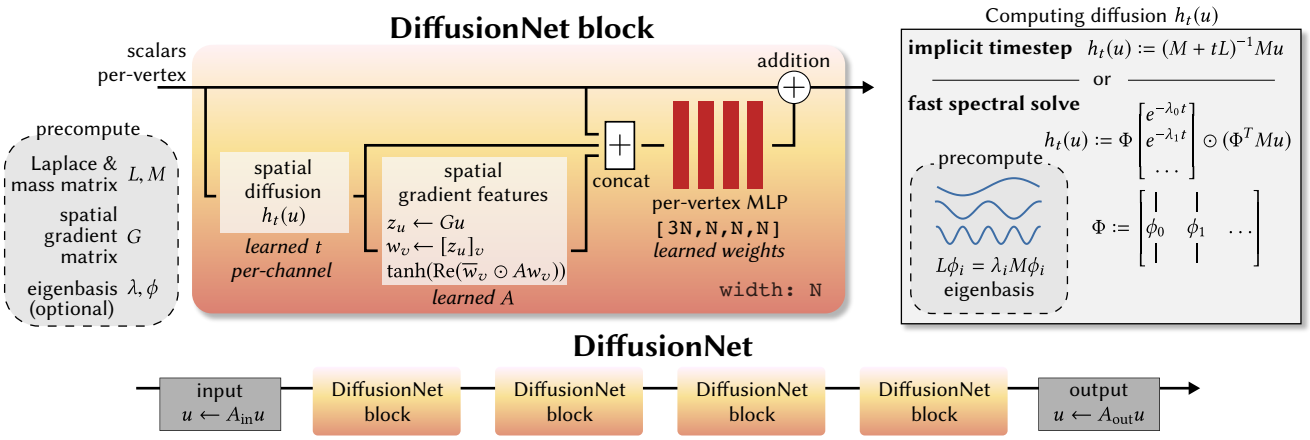


Fig. 7. We present DiffusionNet, a simple and effective architecture for learning on surfaces. It is composed of successive identical DiffusionNet blocks. Each block diffuses every feature for a learned time scale, forms spatial gradient features, and applies a *spatially shared* pointwise MLP at each vertex in a mesh/point cloud/*etc.* These networks achieve state-of-the-art performance on surface learning tasks without any explicit surface convolutions or pooling hierarchies, in part because they automatically optimize for variable spatial support (see *e.g.*, Figure 4).

5 EXPERIMENTS AND ANALYSIS

The same network architecture achieves state-of-the-art results across many tasks, and more importantly offers new and valuable capabilities. See the appendix for additional analyses.

Setup. We use the same basic 4-block DiffusionNet architecture and training procedure for all tasks, varying the network size from a small 32-width (30k parameter) to a large 256-width (1.8M parameter) DiffusionNet according to the scale of the problem. The shape of the first and last linear layers is adapted to the input and output dimension for the problem. MLPs use ReLU activations and optionally dropout after intermediate linear layers. We let "xyz" and "hks" denote networks with positions and heat kernel signatures as input, respectively. All inputs are centered and scaled to be contained in a unit sphere, and heat kernel signatures are sampled at 16 values of t logarithmically spaced on $[0.01, 1]$. We do not use any data augmentation, except random rotations in tasks where positions are used as features yet a rotation-invariant network is desired.

We fit DiffusionNet using the ADAM optimizer with an initial learning rate of 0.001 and a batch size of 1, training for 200 epochs and decaying the learning rate by a factor of 0.5 every 50 epochs. Cross-entropy loss is used for labelling problems. Spectral acceleration is used to evaluate diffusion except where noted, truncated to a $k = 128$ eigenbasis. On point clouds, 30 nearest-neighbors are used to assemble matrices. Test accuracies are measured after the last epoch of training.

Implementation details. Precomputation to assemble matrices and compute the Laplacian eigenbasis for spectral acceleration is performed once as a preprocess on the CPU using SciPy [Lehoucq et al. 1998; Virtanen et al. 2020]. Networks are implemented in PyTorch [Paszke et al. 2019] and evaluated on a single GPU with standard backpropagation. Performance is discussed in Section 5.6; we find that DiffusionNet is very efficient and scalable compared to recent mesh-based learning methods. Code and reproducible experiments are available at github.com/nmwsharp/diffusion-net.

Table 1. DiffusionNet achieves nearly-perfect accuracy classifying 30-class SHREC11 [Lian et al. 2011] while training on just 10 samples per class. Results marked by † are trained and tested on simplified models.

Method	Accuracy
GWCNN [Ezuz et al. 2017]	90.3%
MeshCNN† [Hanocka et al. 2019]	91.0%
HSN† [Wiersma et al. 2020]	96.1%
MeshWalker† [Lahav and Tal 2020]	97.1%
PD-MeshNet† [Milano et al. 2020]	99.1%
HodgeNet† [Smirnov and Solomon 2021]	94.7%
FC† [Mitchel et al. 2021]	99.2%
DiffusionNet - xyz†	99.4%
DiffusionNet - xyz	99.0%
DiffusionNet - hks†	99.5%
DiffusionNet - hks	**99.7%**

5.1 Classification

We first apply DiffusionNet to classify meshes in the SHREC-11 dataset [Lian et al. 2011], which has 30 categories of 20 shapes each. We demonstrate that DiffusionNet learns successfully even in the presence of limited data. As in the other cited results, we train on just 10 samples per class; our results are averaged over 10 trials of the experiment with random training splits. We fit a cross-entropy loss with a label smoothing factor of 0.2 (see discussion in [Goyal et al. 2021]). We use a 32-width DiffusionNet for hks features, and 64-width DiffusionNet for xyz features with rotation augmentation. DiffusionNet achieves the highest reported accuracy when applied directly on the original dataset models, or to the simplified models common in recent mesh-based learning work (Table 1).

Table 2. Accuracy of various mesh and point cloud schemes for RNA segmentation. DiffusionNet achieves state-of-the-art results, in part because it can be applied directly to the raw meshes. "xyz" and "hks" denote networks taking raw coordinates or heat kernel signatures as input, respectively.

	Method	Accuracy
cloud	PointNet++ [Qi et al. 2017a]	74.4%
	PCNN [Atzmon et al. 2018]	78.0%
	SPHNet [Poulenard et al. 2019]	80.1%
	DiffusionNet - hks	84.0%
	DiffusionNet - xyz	**85.0%**
mesh	SplineCNN [Fey et al. 2018]	53.6%
	SurfaceNetworks [Kostrikov et al. 2018]	88.5%
	DiffusionNet - hks	91.0%
	DiffusionNet - xyz	**91.5%**

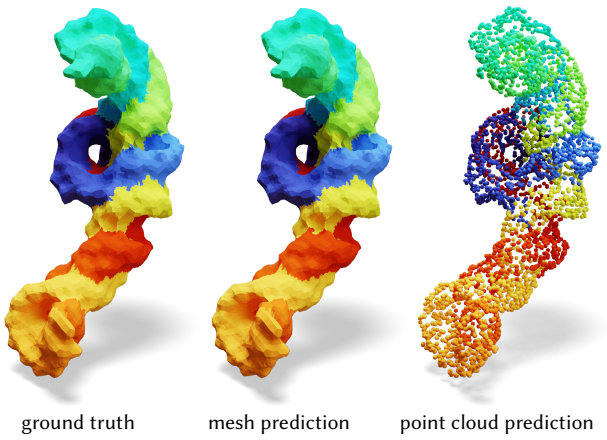


Fig. 8. Segmenting RNA molecules with our method achieves accurate results when applied either directly to meshes or to sampled point clouds.

5.2 Segmentation

Molecular segmentation. We evaluate a 128-width DiffusionNet on the task of segmenting RNA molecules into functional components, using the dataset introduced by Poulenard et al. [2019]. This dataset consists of 640 RNA surface meshes of about 15k vertices each extracted from the Protein Data Bank [Berman et al. 2000], labelled at each vertex according to 259 atomic categories, with a random 80/20 train-test split. We learn these labels directly on the raw meshes, as well as on point clouds of 4096 uniformly-sampled points as in past work Poulenard et al. [2019]. For comparison, we cite point cloud results reported in [Poulenard et al. 2019], and additionally train methods related to ours, SplineCNN [Fey et al. 2018] and a Dirac Surface Network [Kostrikov et al. 2018] on meshes. We also attempted to train MeshCNN [Hanocka et al. 2019] and HSN [Wiersma et al. 2020], but found the former prohibitively expensive, while the latter could not successfully preprocess the data. Our method achieves state-of-the-art accuracy on both the mesh and point cloud variants of the problem (Table 2, Figure 8). Learning directly on the mesh yields greater accuracy, perhaps because no information is lost when sampling a point cloud, and surface structure is preserved.

Human segmentation. We train a 128-width DiffusionNet with dropout to segment the human body parts on the composite dataset of [Maron et al. 2017], containing models from several other human shape datasets [Adobe 2016; Anguelov et al. 2005; Bogo et al. 2014; Giorgi et al. 2007; Vlasic et al. 2008]. Additionally, we cite a variety of reported results from other approaches on this task, as reported by the respective original authors and by Wiersma et al. [2020]. For clarity, we distinguish between variants of this task in past work which used simplified meshes and soft ground truth; more details in Appendix C. Our model is quite effective using both rotation-augmented raw coordinates or heat kernel signatures as input (Table 3).

Table 3. Human part segmentation on the dataset of Maron et al. [2017]. *xyz* and *hks* denote DiffusionNet with positions or heat kernel signatures as input, respectively. The † rows use simplified inputs with a soft ground truth at edges, and the ‡ rows use simplified inputs with hard ground truth at faces; details in Appendix C.

	Method	Accuracy
	GCNN [Masci et al. 2015]	86.4%
	ACNN [Boscaini et al. 2016]	83.7%
	Toric Cover [Maron et al. 2017]	88.0%
	PointNet++ [Qi et al. 2017a]	90.8%
	MDGCNN [Poulenard et al. 2018]	88.6%
	DGCNN [Wang et al. 2019]	89.7%
	SNGC [Haim et al. 2019]	91.0%
	HSN [Wiersma et al. 2020]	91.1%
	MeshWalker [Lahav and Tal 2020]	**92.7%**
	CGConv [Yang et al. 2021]	89.9%
	FC [Mitchel et al. 2021]	92.5%
	DiffusionNet - xyz	90.6%
	DiffusionNet - hks	91.7%
†	MeshCNN [Hanocka et al. 2019]	92.3%
†	MeshWalker [Lahav and Tal 2020]	94.8%
†	DiffusionNet - xyz	**95.5%**
†	DiffusionNet - hks	**95.5%**
‡	PD-MeshNet [Milano et al. 2020]	85.6%
‡	HodgeNet [Smirnov and Solomon 2021]	85.0%
‡	DiffusionNet - xyz	**90.3%**
‡	DiffusionNet - hks	**90.8%**

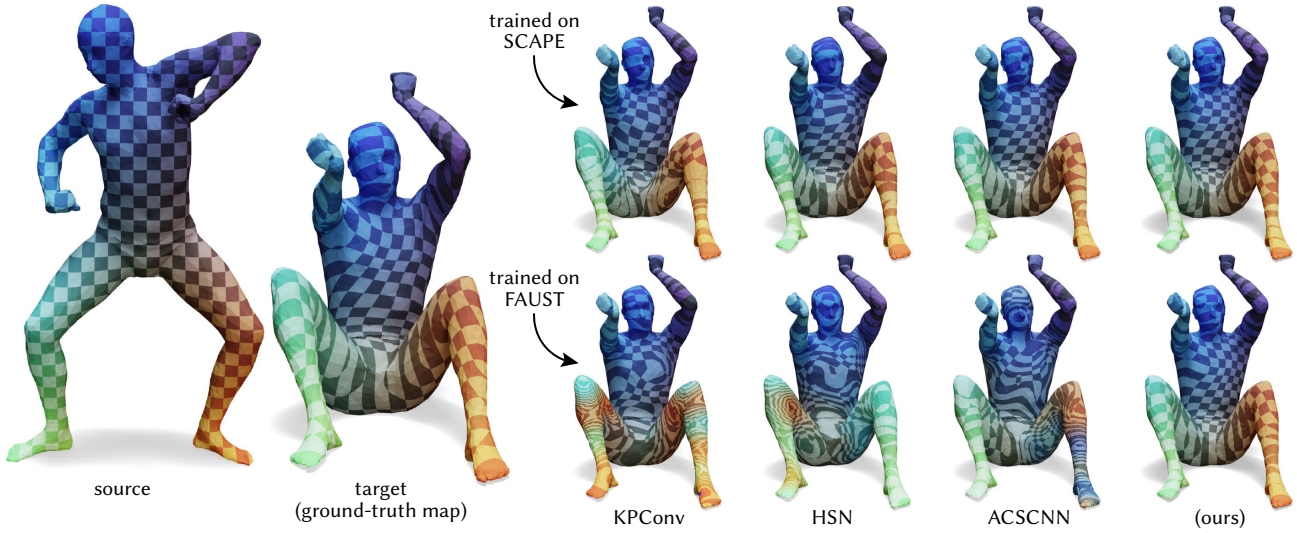


Fig. 9. DiffusionNet is highly effective as a feature extractor for nonrigid correspondence via functional maps, shown here in the supervised setting (Section 5.3). Correspondences are visualized by transferring a texture through the map. All methods yield a visually plausible solution when trained on the same dataset as the query pair (SCAPE, *top row*), but only DiffusionNet yields good results when generalizing after training on a different dataset (FAUST, *bottom row*).

5.3 Functional Correspondence

Functional maps compute a correspondence between a pair of shapes by finding a linear transformation between spectral bases, aligning some set of input features [Ovsjanikov et al. 2012]. Recent work has shown that learned features can improve performance, *e.g.*, Donati et al. [2020]; Litany et al. [2017]. Here we demonstrate that using DiffusionNet as a feature extractor outperforms other recent approaches, yielding to state-of-the-art correspondence results in both the supervised and weakly-supervised variants of the problem. We emphasize that the spectral representation in functional maps is unrelated to the spectral acceleration from Section 3.3.2, which is merely a scheme for evaluating diffusion; DiffusionNet itself does not learn in the spectral domain.

Our experiments follow the setup of Donati et al. [2020], training and evaluating on both SCAPE [Anguelov et al. 2005; Ren et al. 2018], and FAUST [Bogo et al. 2014] including training on one dataset and evaluating on the other. In the *supervised* setting we fit dataset-provided correspondences, and we generate rigid-invariant models by randomly rotating all inputs for training and testing. For the *weakly-supervised* setting, we use the dataset and losses advocated in [Sharma and Ovsjanikov 2020], where rigid alignment of the input is used as weak supervision, without known correspondences. In all cases we extract point-to-point maps between test shapes and evaluate them against ground truth dense correspondences; for simplicity we compare all methods without post-processing the maps, though we also report accuracies after postprocessing with ZoomOut for our method [Melzi et al. 2019]. In addition to citing results obtained using the KPConv feature extractor [Thomas et al. 2019] by Donati et al. [2020], we also train HSN [Wiersma et al. 2020], ACSCNN [Li et al. 2020b], and our own 128-width DiffusionNet with dropout. We also tried MeshCNN [Hanocka et al. 2019], but it proved to be prohibitively expensive at 14hr per epoch.

Table 4. Our approach yields state-of-the-art correspondences when used as a feature extractor for deep functional maps, both in the supervised (*top*, as in Donati et al. [2020]) and the weakly supervised setting (*bottom*, as in Sharma and Ovsjanikov [2020]). The dotted rows apply ZoomOut post-processing to the previous result [Melzi et al. 2019]. X on Y means train on X and test on Y. Values are mean geodesic error ×100 on unit-area shapes.

Method / Dataset	**F**AUST	**S**CAPE	**F** on **S**	**S** on **F**
KPConv [Thomas et al. 2019]	3.1	4.4	11.0	6.0
KPConv - hks	2.9	3.3	10.6	5.5
HSN [Wiersma et al. 2020]	3.3	3.5	25.4	16.7
ACSCNN [Li et al. 2020b]	**2.7**	3.2	8.4	6.0
DiffusionNet - hks	**2.7**	**3.0**	3.8	**3.0**
DiffusionNet - xyz	**2.7**	**3.0**	**3.3**	**3.0**
+ ZoomOut	1.9	2.4	2.4	1.9
WSupFMNet	**3.3**	7.3	11.7	6.2
WSupFMNet + DiffusionNet - xyz	3.8	**4.4**	**4.8**	**3.6**
+ ZoomOut	1.9	2.6	2.7	1.9

As shown in Table 4, DiffusionNet yields state-of-the-art results for non-rigid shape correspondence in the both the supervised and weakly-supervised settings, especially when transferring between datasets. One might wonder whether the improvement in this task truly stems from our DiffusionNet architecture, or from the use of HKS features. As shown in the same table, training KPConv with HKS as features shows DiffusionNet yields significant improvements regardless. Figure 9 visualizes the resulting correspondences on a challenging test pair, where only DiffusionNet achieves a high-quality correspondence when generalizing after training on a different dataset.

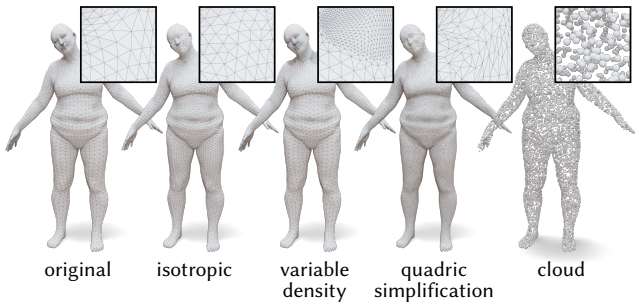


Fig. 10. Examples of the remeshed FAUST test dataset used in Table 5.

Table 5. DiffusionNet automatically retains highly accurate results under changes in meshing and sampling, while many other approaches overfit to mesh connectivity. Here we give correspondence errors on our remeshed FAUST dataset after training on template meshes, measured in mean geodesic distance ×100 after normalizing by the geodesic diameter.

		remeshed/sampled variants			
Method	**orig**	**iso**	**dense**	**qes**	**cloud**
ACSCNN	**0.05**	35.29	19.09	41.15	N/A
SplineCNN	3.51	31.09	27.95	40.43	N/A
HSN	9.57	20.01	24.84	25.40	N/A
PointNet (vertices)	3.83	2.92	3.04	2.67	**2.60**
PointNet (sampled)	9.99	4.25	7.84	4.44	4.13
DGCNN (vertices)	2.44	14.58	15.72	27.00	32.13
DGCNN (sampled)	6.52	4.30	5.57	3.61	2.66
DiffusionNet	0.33	**0.68**	**0.62**	**0.82**	**2.59**

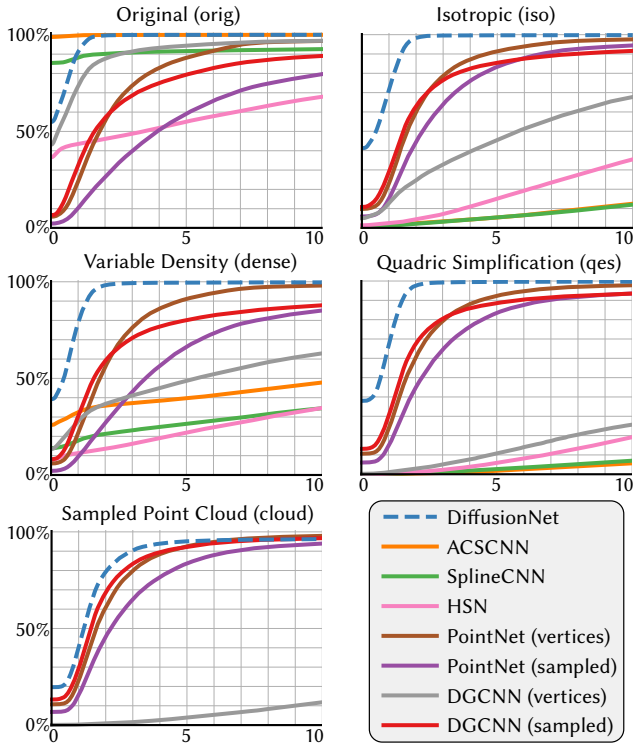


Fig. 11. Accuracy curves for vertex-labelling correspondence on the FAUST dataset, as in Table 5. The first plot gives accuracy on the original test meshes, and the subsequent plots denote testing on our remeshed variants of the test set. For each plot, the x-axis is the geodesic error ×100 after normalizing by geodesic diameter, and the y-axis is the percent of predicted correspondences within that error.

5.4 Discretization Agnostic Learning

A key benefit of DiffusionNet compared to many past approaches is that its outputs are robust to changes in the discretization of the input (*e.g.*, different meshes of the same shape, or a mesh *vs.* a point cloud). This property is essential for practical applications where meshes for inference are likely to be tessellated differently from the training set, *etc.* Other invariants (*e.g.*, rigid invariance) can be encouraged via data augmentation, but for discretization it is impractical to generate augmented inputs across all possible sampling patterns. Below, we demonstrate that DiffusionNet generalizes quite well across discretizations without any special regularization or augmentation, especially compared to recent mesh-based methods.

We study discretization invariance using a popular formulation of the shape correspondence task on the FAUST human dataset [Bogo et al. 2014], where each vertex of a mesh is to be labelled with the corresponding vertex on a template mesh. Importantly, these input meshes are already manually aligned templates, with exactly identical mesh connectivity. Past work has achieved near-perfect accuracy in this problem setup (*e.g.*, [Fey et al. 2018; Li et al. 2020b]), however we suggest that these models primarily overfit to mesh graph structure. In contrast, DiffusionNet learns an accurate and general function of the shape itself, despite the synthetic setup.

To experimentally quantify this effect, we construct a version of the FAUST test set after remeshing with several strategies: *orig* is the original test mesh, *iso* is a uniform isotropic remeshing, *dense* refines the mesh in randomly sampled regions, *qes* first refines the meshes, then applies quadric error simplification [Garland and Heckbert 1997], and *cloud* is a point cloud with normals sampled from the surface (Figure 10). Unrelated remeshings have appeared in [Poulenard and Ovsjanikov 2018; Wiersma et al. 2020], but the procedure therein left large regions of the mesh unchanged. Ground truth for evaluation is defined via nearest-neighbor on the original test mesh. We train on the 3D coordinates of the 80 standard FAUST registered meshes, but evaluate on the remeshed set to mimic the practical scenario where the training set contains meshes tessellated via some particular common strategy, yet the fitted model must be applied to totally different meshes encountered in the wild. We also train several other methods—details in Appendix C.

Table 5 and Figure 11 show how other mesh-based approaches degrade rapidly under remeshing; only DiffusionNet yields accurate correspondences which are largely stable under remeshing and resampling. Some point-based methods avoid the dependence on connectivity, yet do not match the overall accuracy of the surface-based DiffusionNet.

5.5 Transfer Across Representations

Not only are the outputs of DiffusionNet consistent across remeshing and resampling of a shape, but furthermore the same network can be directly applied to different discrete representations. The only geometric data required for DiffusionNet are the Laplacian, mass, and spatial gradient matrices, which are easily constructed for many representations. Because all geometric operations in the network are defined in terms of these standard matrices, fitted network weights retain the same meaning across different representations. This enables us to train on one representation and evaluate on another as seen in Figure 1 and Section 5.4, *cloud*, without any special treatment or fine-tuning. In the future, DiffusionNet opens the door to heterogeneous training sets which intermingle mesh, point cloud, and other surface data from various sources.

5.6 Efficiency and Robustness

Runtime. DiffusionNet requires only standard linear algebra operations for training and inference, and is thus straightforward and efficient on modern hardware. As an example, DiffusionNet with spectral acceleration trains on the 14k-vertex RNA meshes (Section 5.2) in 38ms per input and requires 2.2GB of GPU memory. Preprocessing is performed on the CPU once for each input; for these RNA meshes preprocessing takes 5.4sec and generates 12MB of data each, composed mainly of the Laplacian eigenbasis Φ for spectral acceleration. Table 6 summarizes the runtime performance of DiffusionNet and several other recent methods on the human segmentation task (Section 5.2), including preprocessing, training with gradient computation, and inference. All timings are measured on a 24GB Titan RTX GPU and dual Xeon 5120 2.2Ghz CPUs.

Table 6. Runtimes of DiffusionNet and other mesh-based methods across several different input mesh resolutions. Reported times are for one-time preprocessing (*pre*), a training evaluation with derivatives (*train*), and an inference evaluation (*infer*), each on a single input of the specified size. Entries marked by "—" were infeasibly expensive in time or memory usage. Unlike many recent mesh-based learning methods, DiffusionNet easily trains directly on medium-sized inputs, and even scales to very large meshes.

Method		small 752 vert	medium 10k vert	large 184k vert
DiffusionNet (spectral)	*pre:*	288ms	3.55sec	69.5sec
	train:	19ms	25ms	379ms
	infer:	7ms	10ms	154ms
DiffusionNet (direct)	*pre:*	104ms	—	—
	train:	329ms	—	—
	infer:	81ms	—	—
MeshCNN [Hanocka et al. 2019]	*pre:*	85ms	1.13sec	—
	train:	269ms	2.97sec	—
	infer:	194ms	2.71sec	—
HSN [Wiersma et al. 2020]	*pre:*	905ms	162sec	—
	train:	188ms	1.08sec	—
	infer:	68ms	389ms	—
HodgeNet [Smirnov et al. 2021]	*pre:*	*n/a*	*n/a*	—
	train:	752ms	7.61sec	—
	infer:	645ms	6.87sec	—

Scaling. Most significantly, DiffusionNet's efficiency enables direct learning on common mesh data without dramatic simplification, in contrast to other recent mesh-based schemes. As an example, the segmentation meshes from Maron et al. [2017] have up to 13k vertices, yet recent approaches simplify/downsample to roughly 1k vertices for training, as shown in Figure 2 [Hanocka et al. 2019; Lahav and Tal 2020; Mitchel et al. 2021; Wiersma et al. 2020]. In contrast, our networks easily run at full resolution on this and other datasets, paving the way for adoption in practice and improving accuracy due to preserved details (*e.g.*, in Table 2). We even demonstrate DiffusionNet on a large, 184k vertex raw scan mesh from FAUST—again no special treatment is needed (Table 6, Figure 1).

Robustness. DiffusionNet is also very robust to poor-quality input data; diffusion is a stable smoothing operation, and our method does not require any complex geometry processing operations such as geodesic distance [Masci et al. 2015], edge collapse [Hanocka et al. 2019], parallel transport [Wiersma et al. 2020], or managing pooling hierarchies with upsampling/downsampling. Even the gradient matrix G is the result of a stable least-squares fit. If desired, techniques like the intrinsic Delaunay Laplacian on meshes can be used to further increase robustness [Bobenko and Springborn 2007; Sharp and Crane 2020], though we do not find it necessary in our experiments. We demonstrate in Figure 1 that DiffusionNet can be applied directly to a low-quality, nonmanifold raw scan mesh without any issues.

6 CONCLUSION

We present a new approach for learning on surfaces that is built by using learned diffusion as the main network component, with spatial gradient features to inject directional information. Our method is very efficient to train and evaluate, is robust to changes in sampling, and even generalizes across representations, in addition to achieving state-of-the-art results on a range of tasks.

Limitations. DiffusionNet is designed to leverage the geometric structure of a surface; consequently it is not automatically robust to topological errors or outliers. In fact, diffusion does not allow any communication at all between distinct components of a surface, leading to nonsensical outputs in the presence of spuriously disconnected components (see inset). Subsequent work might mitigate the limitation by combining diffusion with other notions of communication, such as global pooling (*à la* [Qi et al. 2017a]) or edge convolutions over latent nearest-neighbors [Wang et al. 2019].

Fig. 12. Erroneous segmentation results due to disconnected components.

Our networks are intentionally agnostic to local discretization, and thus may not be suited for tasks where one learns some property of the local discrete structure, such as denoising or mesh modification. Finally, although our method discourages overfitting to mesh sampling (Section 5.4), it cannot guarantee to totally eliminate it, and we still observe a small drop in performance when transferring between representations—further investigation will seek to close this gap entirely.

Future work. DiffusionNet can be applied to any surface representation for which a Laplacian matrix and spatial gradients can be constructed. This opens the door to directly learning—and even transferring pretrained networks—on a wide variety of surface representations, from occupancy grids [Caissard et al. 2019] to subdivision surfaces [De Goes et al. 2016]. More broadly, DiffusionNet need not be restricted to explicit surfaces, and could easily be adapted to other geometric domains like volumetric meshes, curve networks, implicit level sets, depth maps, or images. We believe that grounding geometric deep learning in the mathematically and computationally well-established diffusion operation will offer benefits across surface learning and beyond.

ACKNOWLEDGEMENTS

The authors thank Adrien Poulenard and Ruben Wiersma for assistance configuring experiments, and Mark Gillespie for suggesting the mirrored training for Figure 5. Parts of this work were generously supported by the Fields Institute for Research in Mathematical Sciences, the Vector Institute for AI, an NSF Graduate Research Fellowship, ERC Starting Grant No. 758800 (EXPROTEA) the ANR AI Chair AIGRETTE, a Packard Fellowship, NSF CAREER Award 1943123, and gifts from Activision Blizzard, Adobe, Disney, Facebook, and nTopology.

REFERENCES

Adobe. 2016. Adobe Mixamo 3D characters. www.mixamo.com.

Marc Alexa, Philipp Herholz, Maximilian Kohlbrenner, and Olga Sorkine-Hornung. 2020. Properties of Laplace Operators for Tetrahedral Meshes. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 55–68.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*. 408–416.

Matan Atzmon, Haggai Maron, and Yaron Lipman. 2018. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Dominique Beani, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Lió. 2021. Directional Graph Networks. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*. PMLR.

Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15, 6 (2003), 1373–1396.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. 2000. The protein data bank. *Nucleic acids research* 28, 1 (2000), 235–242.

Alexander I Bobenko and Boris A Springborn. 2007. A Discrete Laplace–Beltrami Operator For Simplicial Surfaces. *Disc. & Comp. Geom.* 38, 4 (2007).

Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montufar, Pietro Liò, and Michael M. Bronstein. 2021. Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. 2014. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3794–3801.

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. 2016. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in neural information processing systems*. 3189–3197.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014)*.

Astrid Bunge, Philipp Herholz, Misha Kazhdan, and Mario Botsch. 2020. Polygon laplacian made simple. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 303–313.

Thomas Caissard, David Coeurjolly, Jacques-Olivier Lachaud, and Tristan Roussillon. 2019. Laplace–beltrami operator on digital surfaces. *Journal of Mathematical Imaging and Vision* 61, 3 (2019), 359–379.

Wenming Cao, Zhiyue Yan, Zhiquan He, and Zhihai He. 2020. A comprehensive survey on geometric deep learning. *IEEE Access* 8 (2020), 35929–35949.

Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)* 35, 3 (2008), 1–14.

Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. 2005. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences* 102, 21 (2005), 7426–7431.

Keenan Crane, Fernando De Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH Courses*. 1–126.

Fernando De Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. 2016. Subdivision exterior calculus for geometry processing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

Haowen Deng, Tolga Birdal, and Slobodan Ilic. 2018. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 602–618.

Nicolas Donati, Abhishek Sharma, and M. Ovsjanikov. 2020. Deep Geometric Functional Maps: Robust Feature Learning for Shape Correspondence. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), 8589–8598.

Moshe Eliasof and Eran Treister. 2020. DiffGCN: Graph Convolutional Networks via Differential Operators and Algebraic Multigrid Pooling. In *Advances in neural information processing systems*.

Danielle Ezuz, Justin Solomon, Vladimir G Kim, and Mirela Ben-Chen. 2017. GWCNN: A metric alignment layer for deep shape analysis. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 49–57.

Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. 2019. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 8279–8286.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. 2018. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 869–877.

Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 209–216.

Dvir Ginzburg and Dan Raviv. 2020. Cyclic Functional Mapping: Self-supervised correspondence between non-isometric deformable shapes. *Proc. ECCV* (2020).

Daniéla Giorgi, Silvia Biasotti, and Laura Paraboschi. 2007. Watertight models track. In *Shape Retrieval Contest 2007: CNR—IMATI Via De Marini 6*. Number 16149.

Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. 2019. SpiralNet++: A Fast and Highly Efficient Mesh Convolution Operator. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 0–0.

Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. 2021. Revisiting Point Cloud Shape Classification with a Simple and Effective Baseline. *International Conference on Machine Learning* (2021).

Samuel Greengard. 2020. Geometric Deep Learning Advances Data Science. *Commun. ACM* 64, 1 (Dec. 2020), 13–15.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 2018. 3D-CODED : 3D Correspondences by Deep Deformation. In *ECCV*.

Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2020. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence* (2020).

Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. 2021. Gauge Equivariant Mesh {CNN}s: Anisotropic convolutions on geometric graphs. In *International Conference on Learning Representations*.

Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. 2019. Surface Networks via General Covers. In *Proceedings of the IEEE International Conference on Computer Vision*. 632–641.

Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi. 2020. Cell Complex Neural Networks. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*.

Oshri Halimi, Or Litany, Emanuele Rodola, Alex M Bronstein, and Ron Kimmel. 2019. Unsupervised learning of dense shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4370–4379.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: a network with an edge. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Lasse Hansen, Jasper Diesel, and Mattias P Heinrich. 2018. Multi-kernel diffusion cnns for graph-based learning on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 0–0.

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

Wenchong He, Zhe Jiang, Chengming Zhang, and Arpan Man Sainju. 2020. CurvaNet: Geometric Deep Learning based on Directional Curvature for 3D Shape Analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2214–2224.

Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. 2020. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Niessner. 2018. Spherical CNNs on Unstructured Grids. In *International Conference on Learning Representations*.

Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 2017. 3D Shape Segmentation with Projective Convolutional Networks. In *Proc. CVPR*.

Johannes Klicpera, Stefan Weissenberger, and Stephan Günnemann. 2019. Diffusion Improves Graph Learning. In *NeurIPS*.

Roman Klokov and Victor Lempitsky. 2017. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 863–872.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Transactions on Graphics (ToG)* 32, 4 (2013), 1–10.

Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. 2018. Surface networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2540–2548.

Alon Lahav and Ayellet Tal. 2020. Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–13.

Richard B Lehoucq, Danny C Sorensen, and Chao Yang. 1998. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM.

Matyáš Lerch. 1903. Sur un point de la théorie des fonctions génératrices d'Abel. *Acta mathematica* 27, 1 (1903), 339–351.

Ron Levie, Wei Huang, Lorenzo Bucci, Michael M Bronstein, and Gitta Kutyniok. 2019. Transferability of spectral graph convolutional neural networks. *arXiv preprint arXiv:1907.12972* (2019).

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing* 67, 1 (2018), 97–109.

Bruno Levy. 2006. Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*. IEEE, 13–13.

Maosen Li, Siheng Chen, Ya Zhang, and Ivor W. Tsang. 2020a. Graph Cross Networks with Vertex Infomax Pooling. In *NeurIPS*.

Qinsong Li, Shengjun Liu, Ling Hu, and Xinru Liu. 2020b. Shape correspondence using anisotropic Chebyshev spectral CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14658–14667.

Xianzhi Li, Ruihui Li, Guangyong Chen, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2021. A rotation-invariant framework for deep point cloud analysis. *IEEE Transactions on Visualization and Computer Graphics* (2021).

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems* 31 (2018), 820–830.

Z Lian, A Godil, B Bustos, M Daoudi, J Hermans, S Kawamura, Y Kurita, G Lavoua, and P Dp Suetens. 2011. Shape retrieval on non-rigid 3D watertight meshes. In *Eurographics workshop on 3d object retrieval (3DOR)*.

Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. 2018. A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. 2020. Convolution in the Cloud: Learning Deformable Kernels in 3D Graph Convolution Networks for Point Cloud Analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. 2017. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE International Conference on Computer Vision*. 5659–5667.

Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface Multigrid via Intrinsic Prolongation. *ACM Trans. Graph.* 40, 4, Article 80 (July 2021), 13 pages.

Risheng Liu, Guangyu Zhong, Junjie Cao, Zhouchen Lin, Shiguang Shan, and Zhongxuan Luo. 2016. Learning to diffuse: A new perspective to design pdes for visual analysis. *IEEE transactions on pattern analysis and machine intelligence* 38, 12 (2016), 2457–2471.

Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. 2020. Path Integral Based Convolution and Pooling for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. 16421–16433.

Richard MacNeal. 1949. *The Solution of Partial Differential Equations by Means of Electrical Networks*. Ph.D. Dissertation. Caltech.

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. 2017. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.* 36, 4 (2017), 71–1.

Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*. 37–45.

Daniel Maturana and Sebastian Scherer. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 922–928.

Simone Melzi, Jing Ren, Emanuele Rodolà, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. 2019. ZoomOut: spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics* 38, 6 (2019), 1–14.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 35–57.

Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. 2020. Primal-Dual Mesh Convolutional Neural Networks. *Advances in Neural Information Processing Systems* 33 (2020), 952–963.

Thomas W. Mitchel, Vladimir G. Kim, and Michael Kazhdan. 2021. Field Convolutions for Surface CNNs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 10001–10011.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5115–5124.

Sayan Mukherjee and Qiang Wu. 2006. Estimation of gradients and coordinate covariation in classification. *Journal of Machine Learning Research* 7, Nov (2006), 2481–2514.

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*. 8026–8037.

Yifan Peng, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núñez. 2020. Efficient Long-Range Convolutions for Point Clouds. *arXiv preprint arXiv:2010.05295* (2020).

Michael Perlmutter, Feng Gao, Guy Wolf, and Matthew Hirn. 2020. Geometric Wavelet Scattering Networks on Compact Riemannian Manifolds. In *Proceedings of The First Mathematical and Scientific Machine Learning Conference (Proceedings of Machine Learning Research, Vol. 107)*, Jianfeng Lu and Rachel Ward (Eds.). PMLR, 570–604.

Ulrich Pinkall and Konrad Polthier. 1993. Computing Discrete Minimal Surfaces and their Conjugates. *Exp. Math.* 2, 1 (1993).

Adrien Poulenard and Maks Ovsjanikov. 2018. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.

Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. 2019. Effective rotation-invariant point cnn with spherical harmonics kernels. In *2019 International Conference on 3D Vision (3DV)*. IEEE, 47–56.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1, 2 (2017), 4.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*. 5105–5114.

Jing Ren, Adrien Poulenard, Peter Wonka, and Maks Ovsjanikov. 2018. Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–16.

Jean-Michel Roufosse, Abhishek Sharma, and Maks Ovsjanikov. 2019. Unsupervised deep learning for structured shape matching. In *Proceedings of the IEEE international*

conference on computer vision. 1617–1627.
Raif M Rustamov. 2007. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*. 225–233.
Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. 2017. Exploiting the PANORAMA representation for convolutional neural network classification and retrieval. In *Eurographics Workshop on 3D Object Retrieval*.
Abhishek Sharma and Maks Ovsjanikov. 2020. Weakly Supervised Deep Functional Maps for Shape Matching. *Advances in Neural Information Processing Systems* 33 (2020).
Nicholas Sharp and Keenan Crane. 2020. A Laplacian for Nonmanifold Triangle Meshes. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 69–80.
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. The Vector Heat Method. *ACM Trans. Graph.* 38, 3 (2019).
Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. 2015. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters* 22, 12 (2015), 2339–2343.
Ayan Sinha, Jing Bai, and Karthik Ramani. 2016. Deep learning 3D shape surfaces using geometry images. In *European conference on computer vision*. Springer, 223–240.
Dmitriy Smirnov and Justin Solomon. 2021. HodgeNet: Learning Spectral Geometry on Triangle Meshes. *ACM Trans. Graph.* 40, 4, Article 166 (July 2021), 11 pages.
Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*. 945–953.
Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. 2009. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 1383–1392.
Zhiyu Sun, Ethan Rooke, Jerome Charton, Yusen He, Jia Lu, and Stephen Baek. 2020. Zernet: Convolutional neural networks on arbitrary surfaces via zernike local tangent space estimation. In *Computer Graphics Forum*. Wiley Online Library.
Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. 2019. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*. 6411–6420.
Federico Tombari, Samuele Salti, and Luigi Di Stefano. 2010. Unique signatures of histograms for local surface description. In *European conference on computer vision*. Springer, 356–369.
Bruno Vallet and Bruno Lévy. 2008. Spectral geometry processing with manifold harmonics. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 251–260.
Amir Vaxman, Mirela Ben-Chen, and Craig Gotsman. 2010. A multi-resolution approach to heat kernels on discrete surfaces. In *ACM SIGGRAPH 2010 papers*. 1–10.
Nitika Verma, Edmond Boyer, and Jakob Verbeek. 2018. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2598–2606.
Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272.
Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*. 1–9.
Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 72.
Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)* (2019).
Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. 2016. Dense human body correspondences using convolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 1544–1553.
Ruben Wiersma, Elmar Eisemann, and Klaus Hildebrandt. 2020. CNNs on Surfaces Using Rotation-Equivariant Features. *ACM Trans. Graph.* 39, 4, Article 92 (July 2020), 12 pages.
Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.
Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. 2019. Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 1928–1934.
Kai Xu, Vladimir G Kim, Qixing Huang, Niloy Mitra, and Evangelos Kalogerakis. 2016. Data-driven shape analysis and processing. In *SIGGRAPH ASIA 2016 Courses*. ACM, 4.
Zhangsihao Yang, Or Litany, Tolga Birdal, Srinath Sridhar, and Leonidas Guibas. 2021. Continuous geodesic convolutions for learning on 3d shapes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 134–144.
Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. 2017. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2282–2290.
Hao Zhang, Oliver Van Kaick, and Ramsay Dyer. 2010. Spectral mesh processing. In *Computer graphics forum*, Vol. 29. Wiley Online Library, 1865–1894.
Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. 2019. Rotation invariant convolutions for 3D point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*. IEEE, 204–213.
Yongheng Zhao, Tolga Birdal, Jan Eric Lenssen, Emanuele Menegatti, Leonidas Guibas, and Federico Tombari. 2020. Quaternion Equivariant Capsule Networks for 3D Point Clouds. *ECCV 2020* (2020).
Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh. 2020. Fully Convolutional Mesh Autoencoder using Efficient Spatially Varying Kernels. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. 9251–9262.

A AN ARGUMENT FOR GENERALITY

In Section 3.2, we propose diffusion at various learned timescales followed by a learned pointwise function as the essential components of our method. Although this formulation clearly offers nonlocal support to the pointwise functions, it is not immediately clear how general the resulting function space is. In particular, it is significant to show that this function space includes at least radially symmetric convolutions, a basic building block which has appeared widely in past work. The treatment of radially symmetric convolutions arises because points on surfaces do not generally have canonical tangent coordinates, though it should be noted that recent work has since focused on expanding beyond symmetric filters, and our own method includes gradient features for precisely this purpose. Lemma 1 states that, at least in the flat, continuous setting, this function space is sufficiently general to represent radially symmetric convolutions. Here, we give a full version of this argument and some discussion.

Consider a scalar field $u : \mathbb{R}^2 \to \mathbb{R}$ in the plane. Let $U_r(p) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denote the integral of the field u along the sphere with radius r centered at p, *i.e.* $U_r(p) = \int_{\partial B(p,r)} u(y)dy$. Recall that $u_t(p) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denotes the value of u at p after diffusion for time t. We are interested in $U_r(p)$, because it will enable the evaluation of radially-symmetric convolutions against u. The crux of our argument is to show that $U_r(p)$ can be recovered from $u_t(p)$, which we will formalize by showing the existence of a function transform

$$\mathcal{T} : (\mathbb{R}_{>0} \to \mathbb{R}) \to (\mathbb{R}_{>0} \to \mathbb{R})$$

such that

$$U_r(p) = \mathcal{T}[u_t(p)](r). \tag{7}$$

The heat kernel solution for $u_t(p)$ is given by

$$u_t(p) = \int_{\mathbb{R}^2} u(q) \frac{1}{4\pi t} e^{-\frac{|p-q|^2}{4t}} dq = \int_0^\infty U_r(p) \frac{1}{4\pi t} e^{-\frac{r^2}{4t}} dr, \tag{8}$$

where the second equality moves to a radial integral, recalling that $U_r(p)$ is defined as the integral of u along the sphere of radius r at p. Calculation verifies that this integral has the form of a Laplace transform of $U_r(p)$

$$u_t(p) = \frac{1}{4\pi t} \mathcal{L}[\frac{1}{2\sqrt{r}} U_{\sqrt{r}}(p)](\frac{1}{4t}). \tag{9}$$

Table 7. An ablation study, evaluated on the human segmentation task. Omitting any of the components of our method leads to a significant drop in performance. Manually fixing a non-optimal diffusion time also impairs performance—our learned procedure automatically optimizes a diffusion time for each channel.

Ablation	Accuracy
no diffusion	31.4 %
fixed-time diffusion $t = 0.1$	89.1 %
fixed-time diffusion $t = 0.5$	81.6 %
no gradient features	84.1 %
unlearned gradient features	85.6 %
(full method)	**90.6 %**

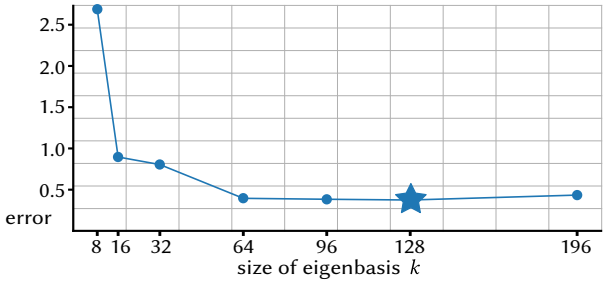


Fig. 13. The effect of varying the size of the truncated basis for spectral diffusion evaluation, measured via error on the FAUST vertex-labelling correspondence from Table 5, *orig*. We use 128 eigenvectors in all experiments.

The Laplace transform is injective [Lerch 1903], which allows us to consider the inverse transform

$$U_r(p) = \mathcal{T}[u_t(p)](r).$$

And in fact, $\mathcal{T}$ will have the form of an inverse Laplace transform, up to reparameterization by $\frac{1}{t}$ and constant coefficients.

Now that we have established the existence of $\mathcal{T}$, it is straightforward to evaluate a radially-symmetric convolution via a pointwise map applied to diffused values. Convolution against any radially symmetric kernel $\alpha(r) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is given by

$$\begin{aligned}(u * \alpha)(p) &= \int_{\mathbb{R}^2} \alpha(|p - q|)u(q)dq \\ &= \int_0^\infty \alpha(r)U_r(p)dr \\ &= \int_0^\infty \alpha(r)\mathcal{T}[u_t(p)](r)dr.\end{aligned} \tag{10}$$

In this sense, the function space defined by diffusion followed by a pointwise map contains the space of radially symmetric convolutions, completing our argument.

Extending this treatment from $\mathbb{R}^2$ to curved manifolds would require a deeper analysis, though the same essential properties hold for diffusion on surfaces. Furthermore, we treat only the continuous setting above, rather than the discrete setting where pointwise maps are approximated via finite-dimensional MLPs, and diffusion is evaluated at a collection of times t. More generally, it would be valuable to extend this analysis to formalize the stability properties of diffusion, *à la* [Kostrikov et al. 2018; Perlmutter et al. 2020]. Nonetheless, we consider this argument to be important evidence that diffusion followed by pointwise functions is an expressive function space, supported by the strong results of our method in practical experiments.

B ANALYSIS

Ablation. To validate the components of our approach, we consider a simple ablation study on the full-resolution human segmentation task from Section 5.2, using rotation-augmented raw coordinates as input. The variant *no diffusion* omits the diffusion layer from each DiffusionNet block, *fixed-time diffusion* manually specifies a diffusion time, *no gradient features* omits the gradient features, and *unlearned gradient features* includes gradient features but omits the learned transformation of gradient vectors A. We observe a noticeable drop in accuracy when omitting any of the components of the method (Table 7). Manually specifying shared, non-optimal diffusion times ($t = 0.1$, $t = 0.5$) yields a network with significantly worse accuracy compared to our learned approach. A key advantage of our learned diffusion is that this time is automatically tuned by the optimization process, individually for each feature channel.

Spectral basis size. When evaluating diffusion with spectral acceleration (Section 3.3.2), increasing the size k of the spectral basis more accurately resolves diffusion at the cost of increased computation. In Figure 13 we vary k for the FAUST vertex-labeling correspondence task as in Table 5, measuring accuracy on the original test set. We find performance degrades significantly with fewer than 64 eigenvectors on this problem, while larger bases offer negligible benefit—our experiments use $k = 128$ eigenvectors as a safe default.

C EXPERIMENT DETAILS

Here we provide additional methodology details for experiments.

Orientation. Figure 5 shows the results of a simple artificial experiment in which we segment the left vs. right side of human models from the FAUST dataset [Bogo et al. 2014] using a purely intrinsic 32-width DiffusionNet with HKS as input. On the original dataset, asymmetric biases–such as a template mesh with asymmetric connectivity–make it unintentionally easy to distinguish left from right. We cancel the effect of these biases by augmenting the dataset with a copy of each mesh that has been mirrored across the left-right axis (preserving orientation by inverting triangles). With a complex-valued A, our network is able to easily distinguish left from right with 99.9% accuracy, despite both a purely intrinsic architecture and intrinsic input features. Restricting to real-valued A removes the effect; the network is unable to disambiguate the symmetry, with a totally random 50.0% accuracy.

Human segmentation. All results are given in Table 3. Past work has used different variations of this dataset, both in terms of the input data and evaluation criteria. The original dataset presented by Maron et al. [2017] contains moderately large meshes of up to 12k vertices, with segmentations labeled per-face, and accuracy is reported as the fraction of faces in the entire test set which were classified correctly. The experiments from Wiersma et al. [2020]

deviate slightly: they remap the ground truth to vertices, and train and test on a subsampling of the vertex set; nonetheless we group these results with the original dataset for the sake of simplicity as they are very similar.

MeshCNN [Hanocka et al. 2019] generated a simplified version of the dataset where the meshes have < 1k vertices, and segmentations have been remapped to edges. Additionally, when reporting test evaluation, a soft ground truth is used allowing for multiple correct segmentation results for edges at the boundary between two regions. For comparison we also apply DiffusionNet to this variant of the task, denoted by † in Table 3—we directly generate a prediction per-edge by averaging per-vertex outputs to edges before applying the final softmax, and evaluate test results against the same soft ground truth. Finally, PD-MeshNet [Milano et al. 2020] generated per-face labels for the MeshCNN simplified models and trained and tested on these *without* any soft ground-truth—we denote this variant by ‡ and again evaluate DiffusionNet with per-face predictions.

Across all variants, DiffusionNet achieves highly accurate performance. Unlike many of these methods, DiffusionNet can easily be trained directly on the original meshes without any special treatment. Even methods which evaluate on full-resolution models may be scalable only due to special pre- and post-processing schemes, which add complexity to adoption in practice—for instance, MeshWalker [Lahav and Tal 2020] trains on simplified meshes then applies an upsampling and smoothing scheme to handle full resolution data.

Discretization agnostic learning. To investigate robustness to discretization on our remeshed FAUST dataset, we train several recent mesh-based and point-based surface learning methods, in addition to our own 256-width DiffusionNet with dropout. For mesh-based methods, we also train SplineCNN [Fey et al. 2018], ACSCNN [Li et al. 2020b], HSN [Wiersma et al. 2020]; we also tried MeshCNN [Hanocka et al. 2019], but found it prohibitively expensive. For point-based methods, we train PointNet [Qi et al. 2017a] and DGCNN [Wang et al. 2019], and consider using both the vertex set as a point set, as well as sampling a random point cloud on the surface, predicting there, then projecting the results back to vertices according to nearest-neighbors. For equivalent comparison, all models are trained with only vertex positions as input (or the constant function, for ACSCNN and SplineCNN), and we augment during training with random rotations about the vertical axis to encourage rotation-invariance. Wherever possible, we mimic the training configuration of the original work or make a best-effort to find suitable parameters for this task. We note that some models perform slightly worse than previously reported results, presumably due to the use of simpler input features or learning in a rotation-invariant setting rather than aligned.

In general, only DiffusionNet learns accurate correspondences which are robust to remeshing and resampling. In particular, ACSCNN still produces nearly perfect results on the original template meshes even in the rotation-invariant setting, but yields essentially random noise after any remeshing. Perhaps unsurprisingly, point-based methods are less prone to overfitting the mesh connectivity (though the DGCNN on vertices still manages to do so), but are still notably less accurate than mesh-based techniques. Figure 11 gives full geodesic error plots corresponding to Table 5.