DiffusionNET: Discretization Agnostic Learning on Surfaces

Til Stotz

Department for informatics - Technische Universität München

Abstract

In the paper DiffusionNET: Discretization Agnostic Learning on Surfaces (2022), the authors Nicholas Sharp, Souhaib Attaiki, Keenan Crane and Maks Ovsjanikov present a novel network architecture for deep learning on 3D surfaces, called DiffusionNET. Its main advantages are its ease of implementation, efficiency and robustness to sampling techniques, different resolutions and applicability to different representations. Three simple ingredients endow the network with those advantages: A standard Multi-Layer Perceptron (MLP), a diffusion method applied to 3D surfaces, and spatial gradient features [5].

1 Introduction

DiffusionNET is an inventive plug-and-play neural network for deep learning on 3D surfaces. Its main advantages are its simple architecture, the robustness to sampling techniques, varying resolutions and different representations and its efficiency, outperforming most other architectures at common benchmarks. The only ingredients are a standard multi-layer perceptron, a diffusion mechanism and spatial gradient features. To emphasize why DiffusionNET is so successful, let's begin by looking at two related architectures that have been widely adopted due to their individual success: PointNET and Dynamic Graph CNN (DGCNN). There are certainly diverse related architectures and different extensions, but related work will be limited to those two in this paper.

PointNET was a pioneering architecture that was directly applicable to point clouds, which made it possible to use such data without a prior transformation to 3D voxel grids. Due to its unified architecture, PointNET could be used for object classification, part segmentation and scene semantic parsing, achieving (at that time) on-par or better than state-of-the-art performance at these tasks. A simplified architecture of PointNET is shown in Fig. 1, taken from [3]. Each point in a point cloud is modeled independently using several shared MLPs, aggregating each output into a global feature using some aggregation function, e.g., a max pooling operation [4]. However, with this architecture two disadvantages become clear. First, the architecture itself can only use point clouds as input. Real-world data comes from a variety of sources, tying a network architecture to a particular representation thus limits the amount of available training data. Second, because the features are learned using independent MLPs, the local structural information between points cannot be captured. Therefore, in non-rigid shape analysis, where the 3D position of points changes between transformations, PointNET cannot achieve significant results.

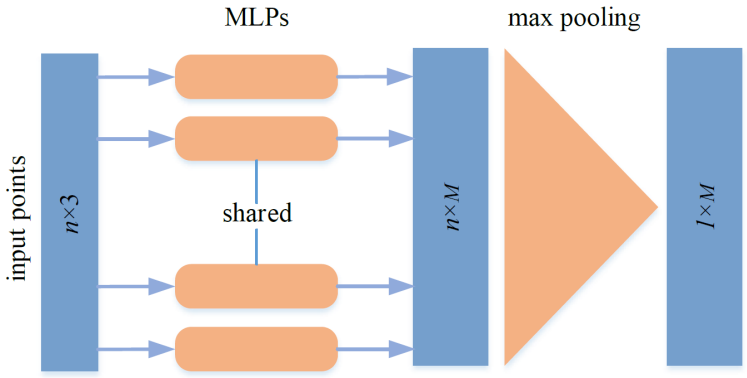


Figure 1: The PointNET architecture consists of several independent MLPs with shared weights, making it impossible to capture local structural information [3].

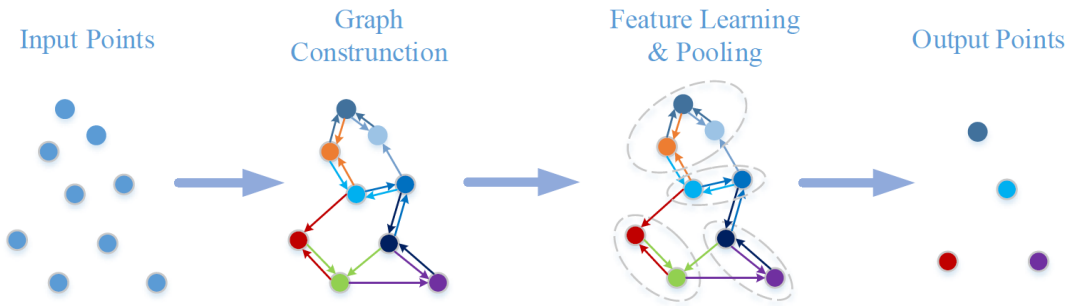


Figure 2: The DGCNN architecture works by creating a graph from a point cloud and learning edge weights between nodes in the graph [8, 3].

One approach at capturing local structural information was developed by Wang et al. in a paper titled "Dynamic Graph CNN for Learning on Point Clouds" (DGCNN). Instead of operating directly on point clouds, the authors proposed to consider each point in a point cloud as a vertex of a graph, shown in Fig. 2. Convolution is then implemented through a standard MLP over spatial neighbors. A subsequent pooling produces a coarser graph, aggregating information from the neighborhood of each point. In the network architecture, this is implemented by an *EdgeConv* operation, which aggregates edge features from connected vertices. The edge features themselves are learned by the network between two point pairs. This procedure allows for the incorporation of local neighborhood information, allows to learn global shape properties and can propagate information over long distances [8, 3]. However, one major drawback becomes clear when applying DGCNN to dense point clouds. Since the network is based on learned edge features between point pairs, increasing the density of a point cloud increases the number of edges. Information propagation across a shape is then slowed down, since a feature essentially needs to traverse more edges than in a shallow point cloud.

In contrast to PointNET, DiffusionNET generalizes its applicability not only to point clouds and meshes, but also captures local structural information, achieving

meaningful results in non-rigid shape analysis. Remarkably, instead of trying to define a convolution and pooling operation like in DGCNN, DiffusionNET propagates information using the technique of diffusion, which is not only a much simpler operation, but also makes information propagation robust against the density of meshes or point clouds. Section 2 explains diffusion and its interaction with the other ingredients in detail.

2 Method description

The DiffusionNET architecture consists of three main ingredients that interact in the overall network architecture. These are a standard multi-layer perceptron, a diffusion method to propagate features, and spatial gradient features to extend the filter space. After a review of these individual ingredients their overall interaction in the network architecture is explained.

Multi-layer Perceptron: The first ingredient of DiffusionNET is a simple multi-layer perceptron. The MLP models a pointwise function $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$, applied independently at every vertex, transforming D scalar features at each one. A standard MLP does not allow to capture the spatial structure of a surface. Therefore, the authors implemented a diffusion mechanism, which will be introduced in the following.

Diffusion: The purpose of diffusion is to spatially propagate features across a mesh or point cloud. Intuitively, it can be thought of as a global smoothing process modeled over time. Diffusion is typically modeled by the heat equation, where the rate of diffusion of a scalar field u is given by its Laplacian:

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

Fig. 3 shows how diffusion operates on a 3D mesh. Beginning at the left, after .01 seconds the point source (yellow star) propagated its features to the neighboring region, shown in blue. Increasing the diffusion time t will increase the distance of propagation. For example, after .5 seconds, the features were propagated across the entire mesh. The key concept of diffusion is simply learning the optimal diffusion time t, which allows the network to learn local, but also totally global support. This is implemented by a diffusion layer, which is shown again in the overall network architecture.

Eq. 1 is used in the continuous setting. When working with meshes and point clouds however, diffusion must be discretized. This is done by replacing the Laplacian Δ by two matrices $L \in \mathbb{R}^{VxV}$ and $M \in \mathbb{R}^{VxV}$, with V being the size of the mesh or point cloud. For meshes, the authors are using the *cotan-Laplace* matrix, where V captures adjacency information and M measures cell ares of vertices. For point clouds, the authors use a different Laplacian form described in [6].

The resulting differential equation can then simply be solved for any diffusion time t. However, the authors observed that using simple Euler methods to solve this equation may not scale well to large problems. Instead, they found spectral acceleration greatly increases performance. First, an eigenvalue decomposition of the matrices L and M is performed, finding the eigenvalues λ_i and stacking all

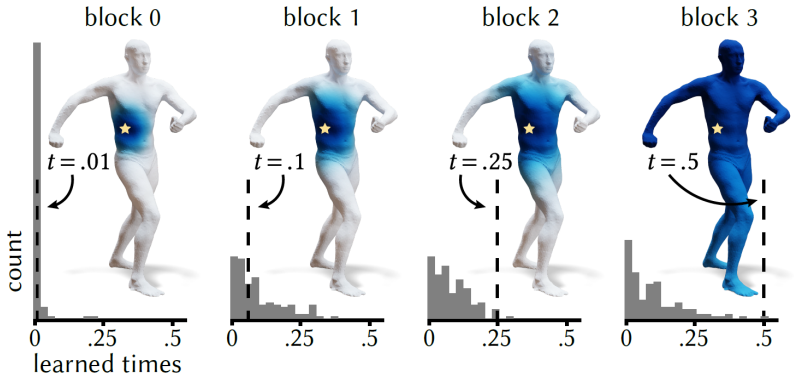


Figure 3: Diffusion is used to propagate features across a mesh. Longer diffusion times correspond to an increasingly global information propagation. The diffusion time is learned by the network.

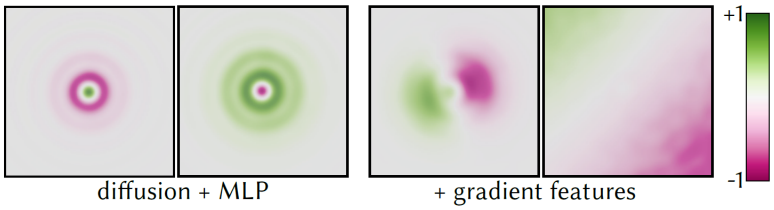


Figure 4: Endowing the network with spatial gradient features enables it to not only realize radially-symmetric filters (left), but also directional filters (right).

eigenvectors to create Φ. Then, computing diffusion in the spectral domain boils down to

$$h_t(u) = \Phi \begin{pmatrix} e^{-\lambda_0 t} \\ e^{-\lambda_1 t} \\ \dots \end{pmatrix} \circ (\Phi^T M u) \tag{2}$$

with $\circ$ denoting the Hadamard product. Intuitively, this corresponds to a projection into the spectral basis (right-hand side multiplication), computation of the diffusion for time t (middle part), and reprojection (left-hand side multiplication by Φ). Eq. 2 shows that computing diffusion can now be expressed by an elementwise exponentiation, speeding up computation time. This form of spectral acceleration enables the network to work with very large meshes or point clouds.

Spatial Gradient Features: Relying only on a diffusion layer would limit the network to radially-symmetric filters only. To extend the space of possible filters, spatial gradient features are used. Fig. 4 shows that endowing the network with these features enables it to implement directional filters. Given a mesh or point cloud of size V, spatial gradient features are found by the following steps:

1. Define a tangent plane at every vertex
2. Project neighboring vertices into the tangent plane
3. Estimate the gradient (e.g. least-squares) in tangent plane

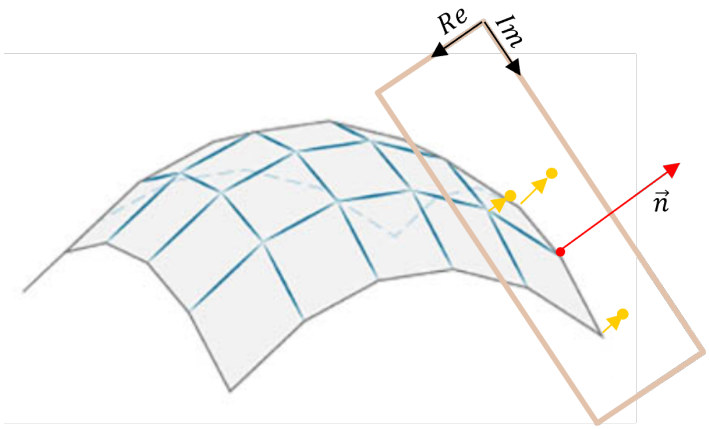


Figure 5: Spatial gradients are computed in the tangent plane of each vertex. The 2D gradients are estimated via least-squares approximation and expressed as complex numbers.

Fig. 5 visualizes these steps. Gradients are expressed by complex numbers, where the axes of the tangent plane correspond to the real and imaginary part. Formally, this can be written as $z_u = Gu$, where $G \in \mathbb{C}^{VxV}$ takes care of projecting each feature channel u and estimating its gradient. After stacking all gradients of all channels to form w_v, scalar features are learned via:

$$g_v(i) = tanh(Re(\sum_{j=1}^{D} \overline{w_v}(i) A_{ij} w_v(j))) \tag{3}$$

Essentially, pairs of gradients are transformed by the coefficient A_{ij}, then their dot product is calculated, exploiting the notation of complex numbers. The outer *tanh* is a simple activation function. Importantly, the coefficient A_{ij} is the only learned parameter in this ingredient and is sufficient to enable the network to work with directional filters.

Network architecture: The three ingredients are implemented in one DiffusionNET block, see Fig. 6. First, the input features are diffused by a diffusion layer, learning the diffusion time t per feature channel. Subsequently, gradients are computed and new features are learned via Eq. 3. Both outputs of these layers and the initial features are then concatenated before they go into the MLP, thus the first layer of the MLP has $3N$ weights. The final output is added to the initial input features to stabilize training. This DiffusionNET block can be repeated as often as suitable, adjusting the last layer depending on the task at hand. Another advantage of this architecture is the ability to precompute many operations. This includes the Laplace and mass matrices, the gradient matrix and, if spectral acceleration is used, the eigenbasis of L and M. For the feature selection, the most basic approach is taking the raw 3D position of vertices. To make the network invariant to orientation-preserving deformations, heat kernel signatures (hks) [7] are preferred.

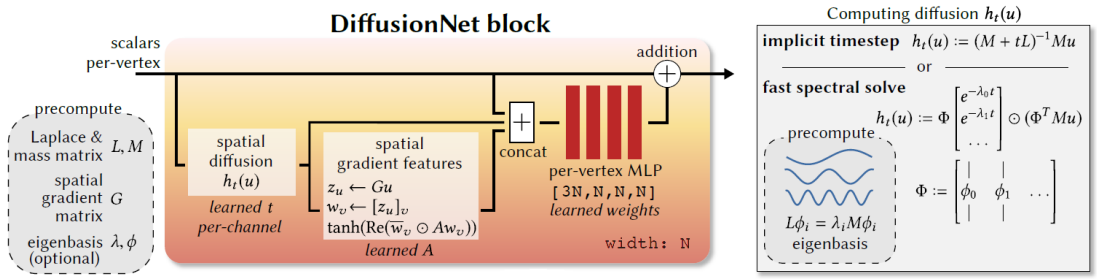
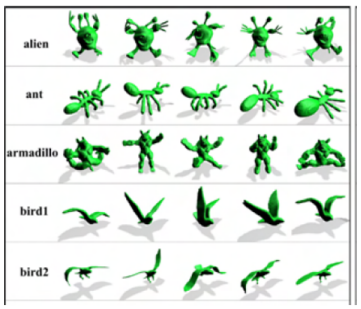


Figure 6: A DiffusionNET block implements all three ingredients: A diffusion layer for spatial propagation, spatial gradient features to realize directional filters and a standard MLP.



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%**

Figure 7: Classification accuracy of different architectures on the SHREC-11 dataset, xyz stands for 3D position as features, hks for heat kernel signatures. DiffusionNET achieves new state-of-the-art accuracy.

3 Experiments and results

DiffusionNET produces state-of-the-art results for many different topics, including classification, molecular- and human segmentation, vertex-labelling correspondence, functional correspondence, runtime and efficiency and many more. Due to the scope of this paper, the presented results are limited to the former three tasks.

Classification: For all benchmarks, the authors use a 4-block DiffusionNET architecture with varying network size depending on the task. The network itself is trained using the ADAM optimizer. Fig 7 shows DiffusionNET's performance at classification on the SHREC-11 dataset. The dataset consists of 30 categories with only 20 shapes per category. Even when using the 3D position as features, DiffusionNET already achieves state-of-the-art performance, improving even more when using heat kernel signatures instead.

Segmentation: Similarily, DiffusionNET achieves state-of-the-art accuracy at segmentation. Fig. 8 visualizes how a 128-width DiffusionNET architecture segments an RNA surface mesh with a size of about 15k vertices. Each vertex has to be labelled correctly to one of 259 atomic categories. Training data stems from the Protein Data Bank [1]. As shown in the figure, DiffusionNET not only produces accurate results for meshes, but is also directly applicable to point clouds.

Discretization Agnostic Learning: To emphasize DiffusionNET's robustness to changes in discretization, different architectures were trained on differently dis-

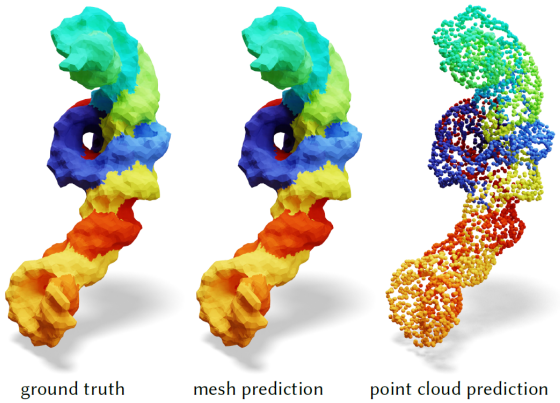


Figure 8: Visualization of a segmentation of an RNA molecule by DiffusionNET. The architecture is applicable to both meshes and point clouds.

cretized inputs and compared results to DiffusionNET. The discretization techniques include isotropic discretization, variable density meshes, quadric simplification and a sampled point cloud. Fig. 9 shows performances of selected architectures at vertex-labelling correspondence on the FAUST dataset [2]. First, DiffusionNET clearly achieves high-quality results independent of the discretization technique, often achieving best accuracy curves. Second, some architectures work better on the original mesh, but their accuracy quickly diminishes for other discretizations. For example, while ACSCNN achieves almost constant 100% accuracy on the original mesh, it only reaches about 50% accuracy on a mesh with variable density. This is true for many other existing architectures, showing that they tend to overfit to the mesh structure. On the contrary, DiffusionNET's accuracy remains almost constant, generalizing better to different discretizations.

4 Discussion

DiffusionNET replaces complex geometry processing operations by a simple and stable diffusion operation. Using diffusion only does not reduce the expressive power of the network for which the authors provide a proof, showing that radially-symmetric convolutions are contained in the function space defined by diffusion followed by a pointwise map (implemented by the MLP). Spatial gradient features inject directional information, extending the space of possible filters. Compared to PointNET, this allows DiffusionNET not only to directly work with different representations, the local spatial information captured by diffusion also enables the network to work with non-rigid transformations. Unlike DGCNN, which can also capture local spatial information but at the cost of bad performance for large meshes, DiffusionNET remains robust against high-resolution inputs and works with 180k vertex meshes.

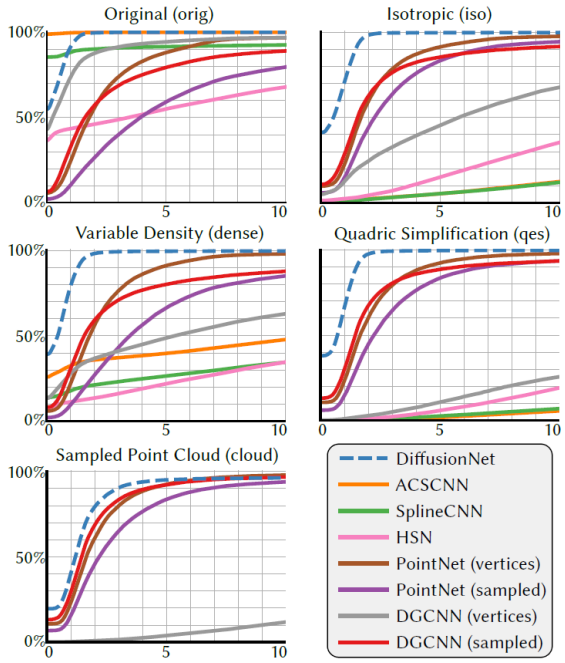


Figure 9: DiffusionNET generalizes across different discretization techniques, achieving almost constant accuracy curves. The x-axis is the geodesic error x100, the y-axis is the percent of predicted correspondences within that error.

However, there do exist some limitations where diffusion struggles to produce good results, shown in Fig. 10. As DiffusionNET leverages the geometric structure, it is prone to errors if this structure is disconnected into single components (Fig. 10 left). In such cases, diffusion does not allow for any communication at all, leading to incorrect segmentation results. Topological merging presents another difficulty. Such merges happen due to small errors in scanning processes to create 3D meshes or point clouds. For example in Fig. 10 in the middle, the hand of the person merges with the chest. In this case, diffusion can now happen directly between the chest area and the hand, whereas in reality diffusion would happen beginning at the chest area and then across the shoulder and arms.

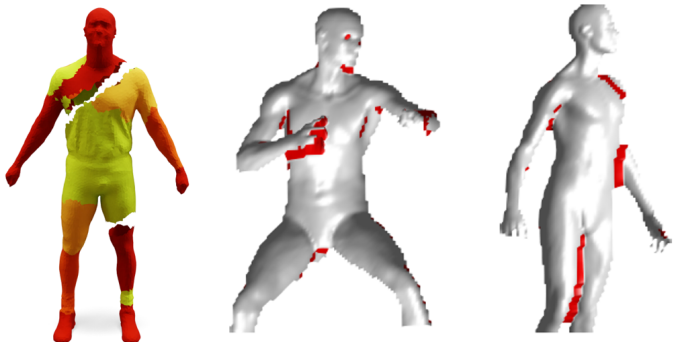


Figure 10: Left: Disconnected components lead to incorrect segmentation, information cannot be propagated. Middle and right: Topological merging results in incorrect diffusion between merged body parts (shown in red).

References

[1] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[2] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3794–3801, Columbus, Ohio, USA, 2014.

[3] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *CoRR*, abs/1912.12033, 2019.

[4] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[5] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusion is all you need for learning on surfaces. *CoRR*, abs/2012.00888, 2020.

[6] Nicholas Sharp and Keenan Crane. A laplacian for nonmanifold triangle meshes. *Computer Graphics Forum*, 39(5):69–80, 2020.

[7] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, 28(5):1383–1392, 2009.

[8] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *CoRR*, abs/1801.07829, 2018.