DiffusionNet: Discretization Agnostic Learning on Surfaces

Louis Martinez
Télécom Paris, MVA
Palaiseau, France
louis.martinez@telecom-paris.fr

Géraud Ilinca
Télécom Paris, MVA
Palaiseau, France
geraud.ilinca@telecom-paris.fr

Hadrien Levéchin
Télécom Paris, MVA
Palaiseau, France
hadrien.levechin@telecom-paris.fr

Keywords

geometric deep learning, representation learning, differential geometry, diffusion on surfaces

1 Abstract

This report proposes an in-depth analysis of the key elements of the DiffusionNet [11], a Deep Learning architecture able to learn robust representations from high resolution geometric data. After addressing the limitations of previous methods to learn on geometric data, we propose an in-depth analysis of the core theoretical notions at stake in the proposed architecture. More specifically, we aim to provide a rigorous yet intuitive explanation of the properties and performances claimed by the authors. Finally, we assess the robustness and expressivity of the representations learned by DiffusionNet blocks through a classification task on a subset of the Objaverse 1.0 dataset.

2 Introduction

2.1 Context

Learning on geometric data serves as a cornerstone for various applications. For instance, in medical imaging, such data is ubiquitous and plays a pivotal role in shape correspondence and segmentation, helping to identify and align anatomical structures. For diagnosis, treatment planning, and surgical precision. In the context of LIDAR (Light Detection and Ranging), shape learning allows for the accurate reconstruction and interpretation of 3D environments encoded as massive point clouds [14], which is crucial for autonomous navigation, urban planning, and environmental monitoring. However, given one object, its digital representation is one discrete instance among the infinite number of possible representations of the underlying continuous object. To address this, it is crucial to develop intrinsic methods that can effectively handle the diversity of real-world data, while being as insensitive as possible to the discretization method.

2.2 Current approaches

Most Machine Learning approaches on geometric data draw inspiration from image processing techniques, particularly from convolutional architectures which effectively capture underlying structure, generalize well and are parameter efficient [8]. While these methods work seamlessly for data defined on regular grids, they are challenging to tailor to irregular surfaces such as 3D meshes. Most approaches attempt to adapt convolution and pooling, which are the two building blocks of CNNs, to these surfaces. However, this is highly non-trivial due to the arising of new challenges, such as the lack of a canonical orientation for local convolutional patches. Thus, early work focused first on voxel grids, as 2D operators can easily be adapted to 3D data. However, those methods turned out to be computationally expensive for a relatively low resolution. Hence, the most successful recent approaches focused first on defining local intrinsic operators [5] and combining them with resampling methods [4], similarly to CNNs for images.

2.3 Limitations

However, they are typically constrained to specific representations (either meshes or point clouds) each with its own pitfalls. Many state-of-the-art architectures overfit on mesh connectivity instead of learning shapes [7]. Also, overly elaborate methods can lead to excessive computational costs and numerical errors. Point cloud architectures [10] are less affected by these issues. Nevertheless, totally losing the connectivity information leads to performance degradation compared to mesh approaches.

2.4 Structure of the report

This report is divided into two main parts:

- In Section 3, we provide an in-depth explanation of the theoretical notions, explaining both the computing efficiency and the agnosticism of the proposed architecture to challenging modifications of the original data, e.g. remeshing or even inferring on point-clouds with a model trained on meshes. Section 3.1 focuses on the heat diffusion process. More precisely we draw an intuitive yet rigorous explanation on why diffusion is particularly well-suited for information sharing between different regions of a geometric structure. We also detail the two numerical schemes proposed by DiffusionNet (implicit timestep and spectral resolution) to implement the so-called DiffusionNet block. Section 3.3 delves into the method used to compute the gradient features used in the method. The explanations provided in the original paper must be completed with some insights from the official implementation of the architecture to have a fine-grained understanding of the role of those features.

- In Section 4 we illustrate the knowledge transfer capabilities of DiffusionNet on a classification task by pretraining a deeper and wider version of the original architecture on a subset of the Objaverse 1.0 dataset and then fine-tune it on the SHREC'11 [9] dataset, used to benchmark model capabilities on classification in the original paper.

3 Proposed method

The architecture of DiffusionNet is based on the observation that diffusion on surfaces facilitates the communication between data points, thus enabling the extraction of semantic information similarly to convolution. Diffusion, however, is inherently independent of the specific sampling method used to generate the geometric data,

making the extracted features less reliant on the object's extrinsic structure and discretization. In the original paper, the authors prove the equivalence between radially symmetric convolution filters - which had been the preferred approach so far to perform convolution-like operations on surfaces - and diffusion followed by a vertex-wise Multi-Layer Perceptron, as explained in Section 3.2. This lemma is key as it highlights one of the main strengths of the proposed approach, which is avoiding the pitfall of choosing convolution kernel size and potential resampling factor without loss of expressivity. The authors even overcome the limitation of radially symmetric filters obtained by adding spatial gradient feature (Section 3.3).

3.1 Heat diffusion on surfaces

3.1.1 Context. In this Section we first motivate the use of heat diffusion as a powerful mechanism to learn on surfaces, and then delve into its concrete setup in DiffusionNet. Diffusion on discrete geometric data has been widely used because it allows us to extract some information about the underlying manifold. A direct implication is that diffusion-based methods are independent from the sampling method on this manifold. All the main use cases of diffusion can roughly be seen as some manipulation of the heat kernel $k_t(x, y)$. This must be understood as the heat transmitted from a source point x to a destination point y after some time t. A quick explanation on how $k_t(x, y)$ is computed is given at the end of this section.

The two main use cases of heat diffusion on surfaces are:

- Shape characterization (up to an isometry) through the Heat Kernel Signature (HKS), introduced by Sun et al. [12] It tracks the evolution of $t \mapsto h_t(x, x), \forall x \in \mathcal{M}$. This signature provides robust multiscale information about the mesh, inherited from the heat kernel properties, while being computationally efficient. The latter aspect is crucial when processing high-quality meshes with reasonable amounts of time and compute resources. That's why a diffusion-based approach is a coherent choice with respect to the claim of the authors to efficiently process highly detailed surfaces without needing any downsampling.

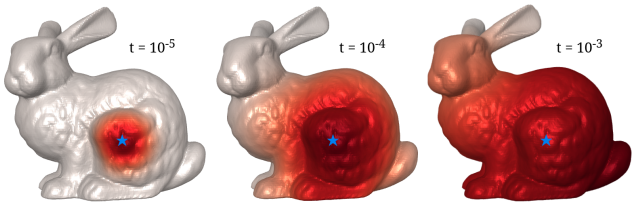

Figure 1: Diffusion of a Dirac on a mesh at different time steps. As time increases, the initial distribution spreads all over the surface to finally reach the average of the initial Dirac. The value initially held at the initial point of the Dirac is thus communicated an increasing neighboring.

- Geodesic distances computation [3]: To further justify why heat diffusion is suitable for shape characterization, the Varadhan's formula provides an explicit link between heat diffusion and geodesic distances:

$$\phi(x, y) = \lim_{t \to 0} \sqrt{-4t \log k_t(x, y)}$$

As the latter are one of the two fundamental elements to fully characterize a 3D shape (with curvature), it is natural to consider diffusion to allow communication between different regions of the surface.

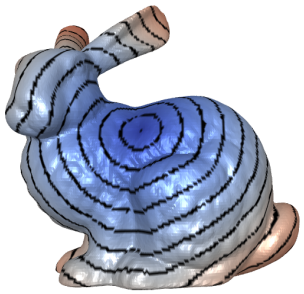

Figure 2: Illustration of geodesic distances computed with diffusion, implemented following the method proposed by Crane et al. Black circles correspond to the isodistance levels from the source point.

3.1.2 Diffusion layer. DiffusionNet takes advantage of the aforementioned points by introducing a diffusion layer. It applies diffusion channel-wise with a learned time. Formally speaking, let assume a feature map of shape (V, K) where V is the number of vertices of the input shape and K the number of features per vertex. In the paper, input features are either 3D coordinates or the HKS of each vertex at different timesteps (typically 16 uniformly spread between 0 and 1). These choices respectively lead to either $K_{in} = 3$ or $K_{in} = 16$. To expand the number of features per vertex, an MLP with shared weights is applied to each vertex. A diffusion block learns one time parameter $t_k, (k \in [\![1, K]\!])$ per input feature. Another advantage of using diffusion is that it allows us to dispense with the choice of hyperparameters related to CNNs, such as the size of kernels and poolings. The values of a given feature channel are then diffused across the whole mesh. At the output of the diffusion block, the value at one vertex is a mixture of the diffused values of the surrounding vertices. So the vertex holds information coming from other vertices. In this sense the paper states that diffusion enables communication between vertices.

We now focus on the actual operations performed by the diffusion block. What is more, we only detail the case of geometries defined as triangular mesh. The reasoning can seamlessly be adapted to other representations of the data, following [1, 2, 7], as mentioned in the paper. Without loss of generality, we focus on a single feature channel, because the principle is the same for the others. Let $u \in \mathbb{R}^V$. The spatial propagation of u follows the heat equation:

$$\frac{\partial u}{\partial t} = \Delta u$$

The solution of this equation is given by the semigroup heat operator $H_t(u) = \exp(t\Delta)u$. As the heat equation is linear, so is H_t. To solve this equation in the case of discretized Riemannian manifolds, some modifications have to be made compared to the continuous case.

- According to the previous remark, the discretized version of H_t is a matrix exponential:

$$H_t(u) = \sum_{n=1}^{\infty} \frac{(t\Delta)^n}{n!}$$

- The Laplacian operator Δ is replaced by the Laplace Beltrami operator with cotangent weights. This choice of weights is ubiquitous in geometry processing and results from a linear interpolation within the triangles defining the mesh. Formally, let $\Delta = -M^{-1}L$ where $L \in \mathbb{R}^{V\times V}$ is the weak Laplace matrix and $M \in \mathbb{R}^{V\times V}$ the diagonal area matrix.

$$M_{ii} = \frac{1}{3} \sum_{f\in\mathcal{N}(v_i)} \text{Area}(f)$$

$$M_{ij} = \begin{cases} \sum_{k\neq i} M_{ik} & \text{if} \quad i = j \\ -\frac{1}{2}\left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right) & \text{if} \quad ij \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{N}(v_i)$ denotes the set of neighbors faces of vertex v_i and $\mathcal{E}$ the set of edges connecting vertices i and j. An important note for later about the numerical computation of the diffused distribution u is that both M and L are sparse.

Now that we have defined the discrete heat semi-group operator, we develop the numerical schemes to efficiently compute the diffused distribution $h_t(u)$ (we reuse the notations of the paper). Before delving into the approaches proposed in the paper, we eliminate the option of computing $h_t(u)$ with power series expansion as it would involve huge computations for poor approximation results. Instead, the paper suggests two simple yet efficient numerical schemes:

- **Backward Euler Method**: This scheme is simply obtained by discretizing in space and time the heat equation:

$$\frac{u_{t+1} - u_t}{dt} = -M^{-1}Lu_{t+1}$$

After rearranging this equation we get

$$(M + dtL)u_{t+1} = Mu_t$$

Thus leading to the formulation of the solution with the heat operator

$$u_{t+1} = h_t(u)$$

with

$$h_t(u) = (M + tL)^{-1}Mu$$

Though this method is straightforward, it requires solving a dense matrix system. But as mentioned before, M and L are sparse. Hence it's not the preferred one as it's highly memory consuming. Note: Backward Euler is key to ensure numerical stability of the solution for a wide range of values of t, unlike the Forward Euler Method.

- **Spectral decomposition**: Similar to the Fourier decomposition, the distribution u can be decomposed on an orthonormal basis of eigenvectors $(\phi_i)_{i\in[[1,k]]}$ corresponding to the k smallest eigenvalues $(\lambda_1, ..., \lambda_k)$ of the Laplace Beltrami operator.

$$u \simeq \sum_{i=}^{k} c_i\phi_i$$

where c_i are the spectral coefficients of u. In this case, computing the diffused distribution at timestep t simply boils down to scaling the coefficients c_i by a factor $e^{-\lambda_i t}$. This scaling factor can be deduced from the expression of the discrete heat kernel:

$$k_t = \sum_{i=1}^{k} e^{-\lambda_i t}\phi_i\phi_i^T \in \mathbb{R}^{V\times V}$$

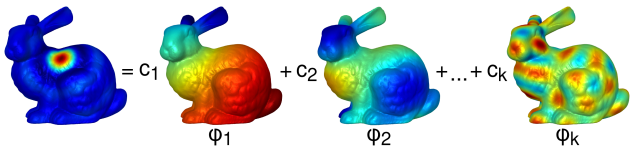

Figure 3: Spectral decomposition of a heat distribution on a mesh. It is projected on an orthonormal eigenbasis of the Laplace Beltrami operator. Similar to the Fourier transform, the eigenvectors can be seen as the equivalent of the sine and cosine functions. λ_i on the other hand, can be interpreted as a vibration mode, or "frequency".

Although diffusion does not require choosing some hyperparameters, the authors claim that they can be as expressive as radially symmetric convolutions, if diffusion is followed by a shared weights vertex-wise MLP. These point emphasize the capabilities of DiffusionNet to seamlessly learn representations from geometric data in a wide range of resolutions.

3.2 Inclusion of radially symmetric convolutions

To justify the authors' claims, we develop an in-depth version of the demonstration of the following lemma proposed in the original paper (stated as in the paper):

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 $u_t(p)$.

Demonstration of the Lemma

Let us revisit the demonstration:

We are working in the context of a manifold. Let p be a point and α a convolution filter with radial symmetry. Then:

$$(u * \alpha)(p) = \int_{\mathcal{M}} \alpha(\|p - q\|)u(q)\, d\mu(q)$$

where u is a function defined on $\mathcal{M}$, $\|p - q\|$ is the geodesic distance between p and q, and $d\mu$ is the measure associated with $\mathcal{M}$.

If we consider the case of surfaces, the manifold is 2-dimensional.

3.2.1 Approximation Hypothesis in $\mathbb{R}^2$. The approximation made in the paper is to restrict the domain to $\mathbb{R}^2$. The limitations of this approximation will be discussed.

In $\mathbb{R}^2$, the convolution with a radially symmetric filter is expressed as:

$$(u * \alpha)(p) = \int_{\mathbb{R}^2} \alpha(\|p - q\|)u(q)\, dq$$

By making the change of variable $v = p - q$:

$$(u * \alpha)(p) = \int_{\mathbb{R}^2} \alpha(\|v\|)u(p - v)\, dv$$

$$(u*\alpha)(p) = \int_{\mathbb{R}^2} \alpha(\|v\|)u(p-v)\, dv = \int_{\mathbb{R}} \int_{\partial B(0,r)} \alpha(r)u(p-y)\, dy\, dr$$

Now,

$$p - \partial B(0, r) = \partial B(p, r)$$

Thus,

$$(u * \alpha)(p) = \int_{\mathbb{R}} \alpha(r) \left[\int_{\partial B(p,r)} u(y)\, dy \right] dr$$

$$(u * \alpha)(p) = \int_{\mathbb{R}} \alpha(r) U_r(p)\, dr$$

where

$$U_r(p) = \int_{\partial B(p,r)} u(y)\, dy$$

According to the heat kernel equation, the next step is to express $U_r(p)$ in terms of $u_t(p)$, which is the result of the diffusion of u at p after a time t.

According to the heat kernel equation, we know that:

$$u_t(p) = \int_{\mathbb{R}^2} k_t(p, y)u(y)\, dy$$

where $k_t(p, y)$ is the diffusion kernel.

The heat kernel in free space $\mathbb{R}^d$ is given by:

$$k_t(p, y) = \frac{1}{(4\pi t)^{d/2}} e^{-\frac{|p-y|^2}{4t}}$$

Thus, in $\mathbb{R}^2$,

$$u_t(p) = \int_{\mathbb{R}^2} \frac{1}{(4\pi t)} e^{-\frac{|p-y|^2}{4t}} u(y)\, dy$$

By making the same change of variable, $v = p - y$,

$$u_t(p) = \int_{\mathbb{R}^2} \frac{1}{(4\pi t)} e^{-\frac{|v|^2}{4t}} u(p - v)\, dv$$

Then,

$$\begin{aligned} u_t(p) &= \int_{\mathbb{R}} \int_{\partial B(0,r)} \frac{1}{(4\pi t)} e^{-\frac{r^2}{4t}} u(p - y)\, dy\, dr \\ &= \int_{\mathbb{R}} \int_{\partial B(p,r)} \frac{1}{(4\pi t)} e^{-\frac{r^2}{4t}} u(y)\, dy\, dr \\ &= \int_{\mathbb{R}} \frac{1}{(4\pi t)} e^{-\frac{r^2}{4t}} \left[\int_{\partial B(p,r)} u(y)\, dy \right] dr \\ &= \int_{\mathbb{R}} \frac{1}{(4\pi t)} e^{-\frac{r^2}{4t}} U_r(p)\, dr \end{aligned}$$

By making the change of variable $s = r^2$,

$$\begin{aligned} u_t(p) &= \int_{\mathbb{R}} \frac{1}{(4\pi t)} e^{-\frac{s}{4t}} U_{\sqrt{s}}(p) \frac{1}{2\sqrt{s}}\, ds \\ &= \frac{1}{(4\pi t)} \int_{\mathbb{R}} \left[\frac{1}{2\sqrt{s}} U_{\sqrt{s}}(p) \right] e^{-s\frac{1}{4t}}\, ds \\ &= \frac{1}{(4\pi t)} \mathcal{L}\left(\frac{1}{2\sqrt{s}} U_{\sqrt{s}}(p) \right) \left(\frac{1}{4t} \right) \end{aligned}$$

Here, $\mathcal{L}$ denotes the Laplace transform.

It is then straightforward to prove that $U_r(p)$ can be expressed in terms of $u_t(p)$ using the injectivity property of the Laplace transform.

We can now write:

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

Thus:

$$(u * \alpha)(p) = \int_{\mathbb{R}} \alpha(r) \mathcal{T}[u_t(p)](r)\, dr$$

We have just shown that the set of symmetric radial convolutions is included in the set of convolutions followed by a point-wise map.

Limitations of the Initial Hypothesis

To recall, the initial hypothesis was that the study could be conducted in $\mathbb{R}^2$ instead of on a Riemannian manifold.

The article mentions that it would ideally be better to work directly on a manifold but justifies the approach in $\mathbb{R}^2$ through experimental evidence and a proof deemed sufficient within this simplified framework.

However, to strengthen the validity of the method, one can argue that a Riemannian manifold is locally Euclidean. Thus, under the assumption of being in a Riemannian manifold, working in a Euclidean space equipped with its distance can be acceptable locally. Furthermore, the symmetric radial convolution used has finite support. If the support of α is sufficiently small, the approximation by $\mathbb{R}^2$ then seems reasonable.

Complexities in the Case of a Manifold

Studying a manifold introduces significant complications. First, one must work with geodesics and define a proper metric for the space. However, under certain reasonable assumptions, it is plausible to replace Euclidean distances with geodesic distances while maintaining a similar proof [13].

Moreover, the solutions to the heat equation on a manifold are far more complex than in $\mathbb{R}^2$ [6]. There is no obvious reason to assume that these equations retain a similar structure or that they directly allow the Laplace transform to emerge.

Assuming these difficulties are overcome and that it is possible to apply the Laplace transform, the injectivity argument should allow us to reach a conclusion similar to that obtained in $\mathbb{R}^2$.

After addressing these issues, the lemma can reasonably be applied in this case. With the experimental results and a solid theoretical proof, the method appears sufficiently robust.

3.3 Spatial Gradient Features

As described earlier, the MLPs and the diffusion mechanism in DiffusionNet are limited to reproducing the behavior of isotropic filters. These filters cannot differentiate between directions, which significantly reduces their expressive power. For instance, in CNNs, learned convolutional filters are highly anisotropic, which is essential for detecting even basic patterns such as lines.

For this reason, the authors introduce local anisotropic filters through the use of spatial gradient features. Since a shape is defined up to a rigid transformation, the operation must remain unchanged under rigid transformations of the 3D space in which the data resides. Furthermore, defining local filters on a surface poses the challenge of the absence of a canonical choice for a coordinate system.

To address these issues, DiffusionNet employs a composition of equivariant operations followed by invariant operations with respect to rigid transformations of space. For a point p in the point cloud or the equipped mesh and its neighborhood N, defined by k-nearest neighbors or points within a certain distance threshold:

(1) An estimate of the normal $\mathbf{n}$ is computed (equivariant under 3D rigid transformations).
(2) A basis is constructed where the first two vectors form a tangent plane basis at p:

$$\mathbf{t}_1 = \frac{\mathbf{n} - \langle n \mid [1,0,0] \rangle}{\|\mathbf{n} - \langle n \mid [1,0,0] \rangle\|},$$
$$\mathbf{t}_2 = \frac{\mathbf{n} \times \mathbf{t}_3}{\|\mathbf{n} \times \mathbf{t}_3\|},$$
$$\mathbf{t}_3 = \mathbf{n}.$$

(equivariant under 3D rigid transformations).
(3) The points in the neighborhood are projected onto the tangent plane at p, and their coordinates are expressed in this new basis (invariant under 3D rigid transformations, up to a rotation of the local basis).
(4) For each feature f_d, the gradient approximation in the tangent plane is computed using least squares by solving an overdetermined linear system:
$\forall \mathbf{n}_i \in N,\ f(\mathbf{n}_i) - f(p) \approx \nabla f_d(p) \cdot (\mathbf{n}_i - p)$
(Equivariant under local basis rotations)
(5) For $i, j \in D^2$, a learnable rotation $R_{i,j}$ is applied, and g_j is computed as:
$g_j = \sum_{i=1}^{D} \langle \nabla f_j(p) \mid R_{i,j} \cdot \nabla f_i(p) \rangle$
(Invariant under local basis rotations)

This approach allows the authors to construct highly expressive directional local filters while maintaining invariance under rigid 3D transformations of the data space.

In practice, for performance reasons, they represent 2D vectors $[a, b]^T$ by complex numbers $a + ib$. The rotations are simply given by the multiplication by a complex $r_{i,j}$ of norm 1. Also, a hyperbolic tangent function may be applied to stabilize training.

The first order approximation of f in neighborhoods of points coupled with the least square approach to derive the gradient value leads to a linear spatial gradient operator, represented by a sparse matrix $G \in \mathbb{C}^{V \times V}$. This approach allow them to perform only once the heavy computations, as for the Laplacian-Beltrami spectral decomposition for the diffusion.

It should be noted that this definition is not invariant to scaling of the data, as the chosen local basis vectors have the same norm as those in the ambient space. Upon inspecting the code, we found that the input data is pre-centered and normalized. The default normalization method (subtracting the mean position followed by division by the max ℓ_2-norm of the translated positions) is invariant under rigid transformations. This normalization further ensures invariance to homotheties relative to the barycenter of the vertices without losing rigid transformation invariance. It is worth noting that this normalization is not strictly invariant under remeshing, but the difference can reasonably be considered negligible.

4 Experiments

In this section, all the results were obtained after training a DiffusionNet on a subset of the Objaverse 1.0 dataset. We used the official implementation of DiffusionNet as a baseline for all our experiments. Specifically, we kept the building blocks of DiffusionNet as is and created a pretraining/fine-tuning pipeline widely inspired from the structure of the already available experiments with the official GitHub repository of the project.

4.1 Model architecture

All the experiments led in the original method used a DiffusionNet architecture with 4 DiffusionNet blocks. Our goal is to learn representations from a larger dataset than the ones tested in the original method, with more variability as well. Hence, we added DiffusionNet blocks and trained the model several times to determine the optimal number of blocks. With more than 6 blocks, the model strongly overfits the training set and validation accuracy drops dramatically from 65% to 30%. The output feature map is then

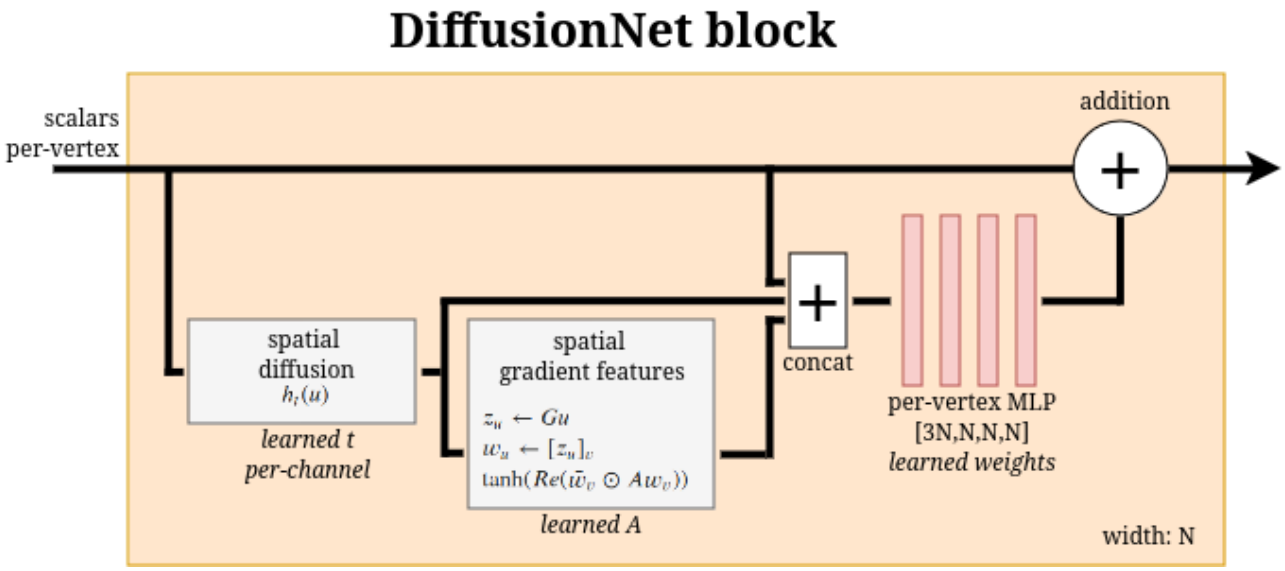

Figure 4: Architecture of a DiffusionNet block. The block is composed of the three aforementioned building blocks. It is important to note that all the operations are independent from the number of vertices, as they're applied channel-wise.

averaged channel-wise and fed to a classification head.

Figure 5: Concatenation of DiffusionNet blocks forming the mpdel used for our experiments.

4.2 Hardware

All the experiments were conducted either on a Nvidia GTX1060 with 6GB VRAM and full-time access or on an A100 with 40GB VRAM but with only part time access.

4.3 Dataset

The raw Objaverse 1.0 dataset presents several pitfalls, making it impossible to use in a consistent way without any heavy preprocessing. There are between 1 and 453 objects per class and 1156 different classes. What's more, the number of vertices per object vary between a few tens and several millions per object. Some classes actually contained high resolution scans of real-world objects where some others are just low poly synthetic geometries. The last pitfall of the dataset lies in the meshing procedure used for some objects. Indeed, the official implementation of the spatial gradient features and Laplacian computation failed for meshes presenting flat triangles for instance. Taking all these issues into account we used the following procedure to build our dataset:

(1) We pre-selected the classes containing more than 100 objects.
(2) The objects containing more than 15000 vertices were removed in each class.
(3) Spatial gradients and Laplacian operators were computed for each object. If the computation fails, the object is dropped.
(4) Classes containing fewer than 40 objects were dropped.
(5) A final manual filtering was performed to drop some classes which would have not allowed good performance, e.g classes where all the objects only vary by their texture (not taken into account here) and not their geometry.

This filtering procedure reduced the dataset to only 42 classes with more than 40 samples each. The different values used for filtering are the consequence of a tradeoff between the available compute power for both operator pre-computation and training, and available storage.

4.4 Training

4.4.1 Pre-training. In order to remain consistent with the experiments led for the original paper, every model was pre-trained for 200 epochs, batch-size of 1, dropout, and a learning rate decay of 0.5 every 50 epochs. However, because there is much more variability in the number of vertices per object, the initial learning rate was set to 10^{-5} (one tenth of the one used in the original paper).

4.4.2 Transfer learning. Only the last DiffusionNet block and the classification head parameters are kept unfrozen. The choice of unfreezing the last DiffusionNet block stems from the empirical assumption that DiffusionNet behaves similarly to a CNN. Consequently, we assume that deeper layers tend to capture semantic information, where the shallow ones capture spatial features. This report does not rigorously justify why this assumption would work better, but based on the similarities between classic convolutional blocks and DiffusionNet block behaviors, this would intuitively allow to partly compensate for the distribution shift between Objaverse1.0 and SHREC'11 datasets.

4.5 Results

We check here that the trained model has similar properties to what was obtained on the original paper. We thus plot the distribution of the learned time parameters in diffusion blocks. Figure 6 shows that the first blocks mainly learned small values of time, where deeper ones learned higher values. This behavior aligns with the observations of the original paper. What is more, it grounds a bit more the intuition of considering the deeper layers as semantic features extractor. Indeed, small diffusion times tend to act locally where with higher diffusion times, the value held by one vertex is diffused across almost all the mesh.

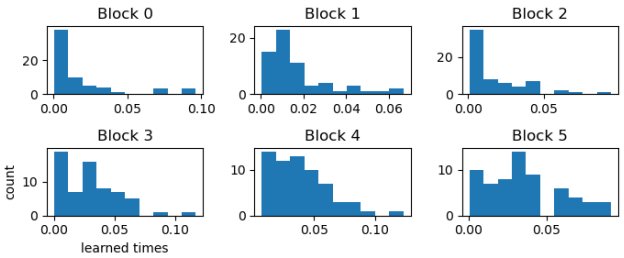

Figure 6: Distribution of the learned times of our model trained on a subset of the Objaverse 1.0 dataset. The obtained results are coherenet with the observations of the paper. Shallow blocks on the one hand, tend to learn small values of t, thus acting local spatial features extractor. On the other hand, deeper blocks have a wider range of times, showing that these blocks capture more semantic information, similarly to convolutional blocks in CNNs.

Table 1 shows the accuracies obtained on our subset of Objaverse 1.0 before retraining the two last layers, and after, on the SHREC'11 dataset. The model whose last blocks were retrained provides extremely poor results. It barely exceeds 10% accuracy on the validation set. This shows that transferring representations learned from one dataset to another is not as straightforward as we would have expected. One of the main limitations of our experiment stems from the heterogeneity of the Objaverse dataset. This point could have been with a larger model model and training on meshes with more vertices.

Model	Training Set	Test Set
Pre-training	78%	60%
Transfer learning	13%	11%

Table 1: Accuracies on training and test sets of the pretrained model, before transfer learning on Objaverse 1.0 an after on SHREC'11.

5 Conclusion

DiffusionNet widely overcame the burden of rigorously defining complex convolutions on meshes through simple yet intuitive mechanisms, which are mathematically grounded. Despite it's apparent simplicity, the model turned out to outperform all the previous approaches on several downstream tasks. In this report we developed those theoretical foundations to explicit intuitions and notions mentioned in the original and that we found to be key in order to have a fine-grained understanding of the whole model.

However, the proposed approach comes with some bottlenecks; the main one mentioned in the article being that the model is not able to handle non connected data.

Finally, despite the claim on the ability of DiffusionNet to learn robust representations, the latter turn out to be not generic enough to be easily transferred from one data distribution to another. It thus shows that using DiffusionNet as a feature extractor is still challenging. It may require some additional considerations, such as adding more classes and objects, and narrowing the range of vertices per objects. Additionally to the dropout implemented within diffusion blocks, we can add additional regularization mechanisms to allow deeper architectures, while avoiding overfitting.

References

[1] Marc Alexa, Philipp Herholz, Maximilian Kohlbrenner, and Olga Sorkine-Hornung. 2020. Properties of Laplace Operators for Tetrahedral Meshes. *Computer Graphics Forum* 39, 5 (Aug. 2020), 55–68. https://doi.org/10.1111/cgf.14068

[2] Astrid Bunge, Philipp Herholz, Misha Kazhdan, and Mario Botsch. 2020. Polygon Laplacian Made Simple. *Computer Graphics Forum* 39, 2 (May 2020), 303–313. https://doi.org/10.1111/cgf.13931

[3] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in Heat. *ACM Transactions on Graphics* 32, 5 (Sept. 2013), 1–11. https://doi.org/10.1145/2516971.2516977 arXiv:1204.6216 [cs].

[4] Michael Garland and Paul S Heckbert. [n. d.]. Surface Simplification Using Quadric Error Metrics. ([n. d.]).

[5] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. 2019. SpiralNet++: A Fast and Highly Efficient Mesh Convolution Operator. https://doi.org/10.48550/arXiv.1911.05856 arXiv:1911.05856 [cs].

[6] Alexander Grigor'yan. 1999. Estimates of heat kernels on Riemannian manifolds. *London Math. Soc. Lecture Note Ser* 273 (1999), 140–225.

[7] Qinsong Li, Shengjun Liu, Ling Hu, and Xinru Liu. 2020. Shape correspondence using anisotropic Chebyshev spectral CNNs. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 14646–14655. https://doi.org/10.1109/CVPR42600.2020.01467

[8] Zhiyuan Li, Yi Zhang, and Sanjeev Arora. 2021. Why Are Convolutional Nets More Sample-Efficient than Fully-Connected Nets? https://doi.org/10.48550/arXiv.2010.08515 arXiv:2010.08515 [cs].

[9] Z Lian, A Godil, B Bustos, M Daoudi, J Hermans, S Kawamura, Y Kurita, G Lavoué, H V Nguyen, R Ohbuchi, Y Ohkita, Y Ohishi, F Porikli, M Reuter, I Sipiran, D Smeets, P Suetens, H Tabia, and D Vandermeulen. [n. d.]. SHREC'11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes. ([n. d.]).

[10] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. https://doi.org/10.48550/arXiv.1612.00593 arXiv:1612.00593 [cs].

[11] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2022. DiffusionNet: Discretization Agnostic Learning on Surfaces. http://arxiv.org/abs/2012.00888 arXiv:2012.00888 [cs].

[12] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. 2009. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Computer Graphics Forum* 28, 5 (July 2009), 1383–1392. https://doi.org/10.1111/j.1467-8659.2009.01515.x

[13] S. Varadhan. 2010. On the Behavior of the Fundamental Solution of the Heat Equation with Variable Coefficients. *Communications on Pure and Applied Mathematics* 20 (05 2010), 431 – 455. https://doi.org/10.1002/cpa.3160200210

[14] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. 2020. Sparse Single Sweep LiDAR Point Cloud Segmentation via Learning Contextual Shape Priors from Scene Completion. https://doi.org/10.48550/arXiv.2012.03762 arXiv:2012.03762 [cs].