# Geometric deep learning on graphs and manifolds using mixture model CNNs

Federico Monti[1*] Davide Boscaini[1*] Jonathan Masci[1,4]
Emanuele Rodolà[1] Jan Svoboda[1] Michael M. Bronstein[1,2,3]
[1]USI Lugano [2]Tel Aviv University [3]Intel Perceptual Computing [4]Nnaisense

## Abstract

*Deep learning has achieved a remarkable performance breakthrough in several fields, most notably in speech recognition, natural language processing, and computer vision. In particular, convolutional neural network (CNN) architectures currently produce state-of-the-art performance on a variety of image analysis tasks such as object detection and recognition. Most of deep learning research has so far focused on dealing with 1D, 2D, or 3D Euclidean-structured data such as acoustic signals, images, or videos. Recently, there has been an increasing interest in* geometric deep learning*, attempting to generalize deep learning methods to non-Euclidean structured data such as graphs and manifolds, with a variety of applications from the domains of network analysis, computational social science, or computer graphics. In this paper, we propose a unified framework allowing to generalize CNN architectures to non-Euclidean domains (graphs and manifolds) and learn local, stationary, and compositional task-specific features. We show that various non-Euclidean CNN methods previously proposed in the literature can be considered as particular instances of our framework. We test the proposed method on standard tasks from the realms of image-, graph- and 3D shape analysis and show that it consistently outperforms previous approaches.*

## 1. Introduction

In recent years, increasingly more fields have to deal with geometric non-Euclidean structured data such as manifolds or graphs. Social networks are perhaps the most prominent example of such data; additional examples include transportation networks, sensor networks, functional networks representing anatomical and functional structure of the brain, and regulatory networks modeling gene expressions. In computer graphics, 3D objects are traditionally modeled as Riemannian manifolds. The success of deep learning methods in many fields has recently provoked a keen interest in *geometric deep learning* [10] attempting to generalize such methods to non-Euclidean structured data.

### 1.1. Related works

**Deep learning on graphs.** The earliest attempts to generalize neural networks to graphs we are aware of are due to Scarselli *et al*. [16, 31]. This work remained practically unnoticed and has been rediscovered only recently [24, 37]. The interest in non-Euclidean deep learning has recently surged in the computer vision and machine learning communities after the seminal work of Bruna *et al*. [11, 18], in which the authors formulated CNN-like [23] deep neural architectures on graphs in the spectral domain, employing the analogy between the classical Fourier transforms and projections onto the eigenbasis of the graph Laplacian operator [33]. In a follow-up work, Defferrard *et al*. [13] proposed an efficient filtering scheme that does not require explicit computation of the Laplacian eigenvectors by using recurrent Chebyshev polynomials. Kipf and Welling [21] further simplified this approach using simple filters operating on 1-hop neighborhoods of the graph. Similar methods were proposed in [3] and [15]. Finally, in the network analysis community, several works constructed graph embeddings [28, 38, 12, 17, 44] methods inspired by the Word2Vec technique [27].

A key criticism of spectral approaches such as [11, 18, 13] is the fact that the spectral definition of convolution is dependent on the Fourier basis (Laplacian eigenbasis), which in turn is domain-dependent. It implies that a spectral CNN model learned on one graph cannot be trivially transferred to another graph with a different Fourier basis, as it would be expressed in a 'different language'.

**Deep learning on manifolds.** In the computer graphics community, we can notice a parallel effort of generalizing deep learning architectures to 3D shapes modeled as manifolds (surfaces). Masci *et al*. [26] proposed the first intrinsic version of convolutional neural networks on manifolds applying filters to local patches represented in geodesic polar coordinates. Boscaini *et al*. [7] used anisotropic heat kernels [1] as an alternative way of extracting intrinsic patches

*Equal contribution

1063-6919/17 $31.00 © 2017 IEEE
DOI 10.1109/CVPR.2017.576

on manifolds. In [6], the same authors proposed a CNN-type architecture in the spatio-frequency domain using the windowed Fourier transform formalism [34]. Sinha *et al*. [35] used geometry images representation to obtain a Euclidean parametrization of 3D shapes on which standard CNNs can be applied.

The key advantage of spatial techniques is that they generalize across different domains, which is a crucial property in computer graphics applications (where a CNN model can be trained on one shape and applied to another one). However, while spatial constructions such as anisotropic heat kernels have a clear geometric interpretation on manifolds, their interpretation on general graphs is somewhat elusive.

### 1.2. Main contribution

In this paper, we present *mixture model networks* (MoNet), a general framework allowing to design convolutional deep architectures on non-Euclidean domains such as graphs and manifolds. Our approach follows the general philosophy of spatial-domain methods such as [26, 7, 3], formulating convolution-like operations as template matching with local intrinsic 'patches' on graphs or manifolds. The key novelty is in the way in which the patch is extracted: while previous approaches used fixed patches, e.g. in geodesic or diffusion coordinates, we use a parametric construction. In particular, we show that patch operators can be constructed as a function of local graph or manifold pseudo-coordinates, and study a family of functions represented as a mixture of Gaussian kernels. Such a construction allows to formulate previously proposed Geodesic CNN (GCNN) [26] and Anisotropic CNN (ACNN) [7] on manifolds or GCN [21] and DCNN [3] on graphs as particular instances of our approach.

Among applications on which we exemplify our approach are classical problems from the realms of image-, graph- and 3D- shape analysis. In the first class of problems, the task is to classify images, treated as adjacency graphs of superpixels. In the second class of problems, we perform vertex-wise classification on a graph representing a citation network of scientific papers. Finally, we consider the problem of finding dense intrinsic correspondence between 3D shapes, treated as manifolds. In all the above problems, we show that our approach consistently outperforms previously proposed non-Euclidean deep learning methods. Due to space limitations, additional results and figures are presented in the Supplementary Materials.

## 2. Deep learning on graphs

Let $\mathcal{G} = (\{1,\dots,n\}, \mathcal{E}, \mathbf{W})$ be an undirected weighted graph, represented by the *adjacency matrix* $\mathbf{W} = (w_{ij})$, where $w_{ij} = w_{ji}$, $w_{ij} = 0$ if $(i,j) \notin \mathcal{E}$ and $w_{ij} > 0$ if $(i,j) \in \mathcal{E}$. The (unnormalized) *graph Laplacian* is an $n \times n$ symmetric positive-semidefinite matrix $\boldsymbol{\Delta} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = \operatorname{diag}\left(\sum_{j\neq i} w_{ij}\right)$ is the *degree matrix*.

The Laplacian has an eigendecomposition $\boldsymbol{\Delta} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\top$, where $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \dots \boldsymbol{\phi}_n)$ are the orthonormal eigenvectors and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1,\dots,\lambda_n)$ is the diagonal matrix of corresponding eigenvalues. The eigenvectors play the role of Fourier atoms in classical harmonic analysis and the eigenvalues can be interpreted as frequencies. Given a signal $\mathbf{f} = (f_1,\dots,f_n)^\top$ on the vertices of graph $\mathcal{G}$, its *graph Fourier transform* is given by $\hat{\mathbf{f}} = \boldsymbol{\Phi}^\top \mathbf{f}$. Given two signals $\mathbf{f}, \mathbf{g}$ on the graph, their *spectral convolution* can be defined as the element-wise product of the Fourier transforms,

$$\mathbf{f} \star \mathbf{g} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \mathbf{f}) \circ (\boldsymbol{\Phi}^\top \mathbf{g}) = \boldsymbol{\Phi} \operatorname{diag}(\hat{g}_1,\dots,\hat{g}_n)\hat{\mathbf{f}}, \quad (1)$$

which corresponds to the property referred to as the *Convolution Theorem* in the Euclidean case.

**Spectral CNN.** Bruna *et al*. [11] used the spectral definition of convolution (1) to generalize CNNs on graphs, with a spectral convolutional layer of the form

$$\mathbf{f}_l^{\text{out}} = \xi\left(\sum_{l'=1}^{p} \boldsymbol{\Phi}_k \hat{\mathbf{G}}_{l,l'} \boldsymbol{\Phi}_k^\top \mathbf{f}_{l'}^{\text{in}}\right). \quad (2)$$

Here the $n \times p$ and $n \times q$ matrices $\mathbf{F}^{\text{in}} = (\mathbf{f}_1^{\text{in}},\dots,\mathbf{f}_p^{\text{in}})$ and $\mathbf{F}^{\text{out}} = (\mathbf{f}_1^{\text{out}},\dots,\mathbf{f}_q^{\text{out}})$ represent respectively the $p$- and $q$-dimensional input and output signals on the vertices of the graph, $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1,\dots,\boldsymbol{\phi}_k)$ is an $n \times k$ matrix of the first eigenvectors, $\hat{\mathbf{G}}_{l,l'} = \operatorname{diag}(\hat{g}_{l,l',1},\dots,\hat{g}_{l,l',k})$ is a $k \times k$ diagonal matrix of spectral multipliers representing a learnable filter in the frequency domain, and $\xi$ is a nonlinearity (e.g. ReLU) applied on the vertex-wise function values. The analogy of pooling in this framework is a graph coarsening procedure, which, given a graph with $n$ vertices, produces a graph with $n' < n$ vertices and transfers signals from the vertices of the fine graph to those of the coarse one.

While conceptually important, this framework has several major drawbacks. First, the spectral filter coefficients are *basis dependent*, and consequently, a spectral CNN model learned on one graph cannot be applied to another graph. Second, the computation of the forward and inverse graph Fourier transform incurs expensive $\mathcal{O}(n^2)$ multiplication by the matrices $\boldsymbol{\Phi}, \boldsymbol{\Phi}^\top$, as there is no FFT-like algorithms on general graphs. Third, there is no guarantee that the filters represented in the spectral domain are localized in the spatial domain; assuming $k = O(n)$ eigenvectors of the Laplacian are used, a spectral convolutional layer requires $pqk = O(n)$ parameters to train.

**Smooth Spectral CNN.** In a follow-up work, Henaff *et al*. [18] argued that *smooth* spectral filter coefficients result in spatially-localized filters and used parametric filters of the

form

$$\hat{g}_i = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda_i), \tag{3}$$

where $\beta_1(\lambda), \ldots, \beta_r(\lambda)$ are some fixed interpolation kernels, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$ are the interpolation coefficients. In matrix notation, the filter is expressed as $\mathrm{diag}(\hat{\mathbf{G}}) = \mathbf{B}\boldsymbol{\alpha}$, where $\mathbf{B} = (b_{ij}) = (\beta_j(\lambda_i))$ is a $k \times r$ matrix. Such a parametrization results in filters with a number of parameters constant in the input size $n$.

**Chebyshev Spectral CNN (ChebNet).** In order to alleviate the cost of explicitly computing the graph Fourier transform, Defferrard *et al*. [13] used an explicit expansion in the Chebyshev polynomial basis to represent the spectral filters

$$g_{\boldsymbol{\alpha}}(\boldsymbol{\Delta}) = \sum_{j=0}^{r-1} \alpha_j T_j(\tilde{\boldsymbol{\Delta}}) = \sum_{j=0}^{r-1} \alpha_j \boldsymbol{\Phi} T_j(\tilde{\boldsymbol{\Lambda}}) \boldsymbol{\Phi}^\top, \tag{4}$$

where $\tilde{\boldsymbol{\Delta}} = 2\lambda_n^{-1}\boldsymbol{\Delta} - \mathbf{I}$ is the rescaled Laplacian such that its eigenvalues $\tilde{\boldsymbol{\Lambda}} = 2\lambda_n^{-1}\boldsymbol{\Lambda} - \mathbf{I}$ are in the interval $[-1, 1]$, $\boldsymbol{\alpha}$ is the $r$-dimensional vector of polynomial coefficients parametrizing the filter, and

$$T_j(\lambda) \quad = \quad 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda), \tag{5}$$

denotes the Chebyshev polynomial of degree $j$ defined in a recursive manner with $T_1(\lambda) = \lambda$ and $T_0(\lambda) = 1$.

Such an approach has several important advantages. First, it does not require an explicit computation of the Laplacian eigenvectors. Due to the recursive definition of the Chebyshev polynomials, the computation of the filter $g_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f}$ entails applying the Laplacian $r$ times, resulting in $\mathcal{O}(rn)$ operations. Second, since the Laplacian is a local operator affecting only 1-hop neighbors of a vertex and accordingly its $(r-1)$st power affects the $r$-hop neighborhood, the resulting filters are localized.

**Graph convolutional network (GCN).** Kipf and Welling [21] considered the construction of [13] with $r = 2$, which, under the additional assumption of $\lambda_n \approx 2$, and $\alpha = \alpha_0 = -\alpha_1$ yields single-parametric filters of the form $g_\alpha(\mathbf{f}) = \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}$. Such a filter is numerically unstable since the maximum eigenvalue of the matrix $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is 2; a renormalization

$$g_\alpha(\mathbf{f}) \quad = \quad \alpha \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{W}} \tilde{\mathbf{D}}^{-1/2} \mathbf{f}, \tag{6}$$

with $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathrm{diag}(\sum_{j \neq i} \tilde{w}_{ij})$ is introduced by the authors in order to cure such problem and allow multiple convolutional levels to be casted one after the other.

**Diffusion CNN (DCNN).** A different spatial-domain method was proposed by Atwood and Towsley [3], who considered a diffusion (random walk) process on the graph. The transition probability of a random walk on a graph is given by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. Different features are produced by applying diffusion of different length (the powers $\mathbf{P}^0, \ldots, \mathbf{P}^{r-1}$),

$$\mathbf{f}_{l,j}^{\mathrm{out}} = \xi(w_{lj}\mathbf{P}^j \mathbf{f}_l^{\mathrm{in}}),$$

where the $n \times p$ and $n \times pr$ matrices $\mathbf{F}^{\mathrm{in}} = (\mathbf{f}_1^{\mathrm{in}}, \ldots, \mathbf{f}_p^{\mathrm{in}})$ and $\mathbf{F}^{\mathrm{out}} = (\mathbf{f}_{1,1}^{\mathrm{out}}, \ldots, \mathbf{f}_{p,r}^{\mathrm{out}})$ represent the $p$- and $pr$-dimensional input and output signals on the vertices of the graph and $\mathbf{W} = (w_{lj})$ is the $p \times r$ matrix of weights.

## 3. Deep learning on manifolds

Let $\mathcal{X}$ be a $d$-dimensional differentiable manifold, possibly with boundary $\partial\mathcal{X}$. Around point $x \in \mathcal{X}$, the manifold is homeomorphic to a $d$-dimensional Euclidean space referred to as the *tangent space* and denoted by $T_x\mathcal{X}$. An inner product $\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$ depending smoothly on $x$ is called the *Riemannian metric*. In the following, we denote by $f : \mathcal{X} \to \mathbb{R}$ smooth real functions (scalar fields) on the manifold. In shape analysis, 3D shapes are modeled as 2-dimensional manifolds (surfaces), representing the boundaries of 3D volumes.

**Geodesic CNN (GCNN).** Masci *et al*. [26] introduced a generalization of CNNs on 2-dimensional manifolds, based on the definition of a local charting procedure in geodesic polar coordinates [22]. Such a construction, named the *patch operator*

$$(D(x)f)(\rho, \theta) = \int_{\mathcal{X}} w_{\rho,\theta}(x, y) f(y) dy$$

maps the values of the function $f$ at a neighborhood of the point $x \in \mathcal{X}$ into the local polar coordinates $\rho, \theta$. Here $dy$ denotes the area element induced by the Riemannian metric, and $w_{\rho,\theta}(x, y)$ is a weighting function localized around $\rho, \theta$ (see examples in Figure 1). $D(x)f$ can be regarded as a *patch* on the manifold; the *geodesic convolution*

$$(f \star g)(x) = \max_{\Delta\theta \in [0, 2\pi)} \int_0^{2\pi} \int_0^{\rho_{\max}} g(\rho, \theta + \Delta\theta)(D(x)f)(\rho, \theta) d\rho d\theta,$$

can be thought of as matching a template $g(\rho, \theta)$ with the extracted patch at each point, where the maximum is taken over all possible rotations of the template in order to resolve the origin ambiguity in the angular coordinate. The geodesic convolution is used to define an analogy of a traditional convolutional layer in GCNN, where the templates $g$ are learned.

**Anisotropic CNN (ACNN).** Boscaini *et al*. [7] considered the *anisotropic diffusion* equation on the manifold

$$f_t(x, t) = -\mathrm{div}_{\mathcal{X}}(\mathbf{A}(x) \nabla_{\mathcal{X}} f(x, t)), \tag{7}$$

where $\nabla_{\mathcal{X}}$ and $\mathrm{div}_{\mathcal{X}}$ denote the *intrinsic gradient* and *divergence*, respectively, $f(x, t)$ is the temperature at point

Table 1. Several CNN-type geometric deep learning methods on graphs and manifolds can be obtained as a particular setting of the proposed framework with an appropriate choice of the pseudo-coordinates and weight functions in the definition of the patch operator. $x$ denotes the reference point (center of the patch) and $y$ a point within the patch. $\mathbf{x}$ denotes the Euclidean coordinates on a regular grid. $\bar{\alpha}, \bar{\sigma}_\rho, \bar{\sigma}_\theta$ and $\bar{\mathbf{u}}_j, \bar{\theta}_j, j = 1, \dots, J$ denote fixed parameters of the weight functions.

| Method | Pseudo-coordinates | $\mathbf{u}(x,y)$ | Weight function $w_j(\mathbf{u}), j = 1, \dots, J$ |
|---|---|---|---|
| CNN [23] | Local Euclidean | $\mathbf{x}(x,y) = \mathbf{x}(y) - \mathbf{x}(x)$ | $\delta(\mathbf{u} - \bar{\mathbf{u}}_j)$ |
| GCNN [26] | Local polar geodesic | $\rho(x,y), \theta(x,y)$ | $\exp(-\frac{1}{2}(\mathbf{u} - \bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u} - \bar{\mathbf{u}}_j))$ |
| ACNN [7] | Local polar geodesic | $\rho(x,y), \theta(x,y)$ | $\exp(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} \begin{pmatrix} \bar{\alpha} & \\ & 1 \end{pmatrix} \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u})$ |
| GCN [21] | Vertex degree | $\deg(x), \deg(y)$ | $\left(1 - \lvert 1 - \frac{1}{\sqrt{u_1}} \rvert\right) \left(1 - \lvert 1 - \frac{1}{\sqrt{u_2}} \rvert\right)$ |
| DCNN [3] | Transition probability in $r$ hops | $p^0(x,y), \dots, p^{r-1}(x,y)$ | $\mathrm{id}(u_j)$ |

$x$ and time $t$, and the *conductivity tensor* $\mathbf{A}(x)$ (operating on the gradient vectors in the tangent space $T_x\mathcal{X}$) allows to model heat flow that is position- and direction-dependent. In particular, they used the $2 \times 2$ tensor

$$\mathbf{A}_{\alpha\theta}(x) = \mathbf{R}_\theta(x) \begin{pmatrix} \alpha & \\ & 1 \end{pmatrix} \mathbf{R}_\theta^\top(x), \tag{8}$$

where matrix $\mathbf{R}_\theta$ is a rotation by $\theta$ in the tangent plane w.r.t. the maximal curvature direction, and the parameter $\alpha > 0$ controls the degree of anisotropy (isotropic diffusion is obtained for $\alpha = 1$). Using as initial condition $f(x,0)$ a point source of heat at $x$, the solution to the heat equation (7) is given by the *anisotropic heat kernel* $h_{\alpha\theta t}(x,y)$, representing the amount of heat that is transferred from point $x$ to point $y$ at time $t$. By varying the parameters $\alpha$, $\theta$ and $t$ (controlling respectively the elongation, orientation, and scale of the kernel) one obtains a collection of kernels that can be used as weighting functions in the construction of the patch operator (see examples in Figure 1). This gives rise to an alternative charting to the geodesic patches of GCNN, more robust to geometric noise, and more efficient to compute.

Both GCNN and ACNN operate in the spatial domain and thus do not suffer from the inherent inability of spectral methods to generalize across different domains. These methods were shown to outperform all the known handcrafted approaches for finding intrinsic correspondence between deformable shapes [26, 7], a notoriously hard problem in computer graphics.

## 4. Our approach

The main contribution of this paper is a generic spatial-domain framework for deep learning on non-Euclidean domains such as graphs and manifolds. We use $x$ to denote, depending on context, a point on a manifold or a vertex of a graph, and consider points $y \in \mathcal{N}(x)$ in the neighborhood of $x$. With each such $y$, we associate a $d$-dimensional vector of *pseudo-coordinates* $\mathbf{u}(x,y)$. In these coordinates, we define a weighting function (kernel) $\mathbf{w}_{\boldsymbol{\Theta}}(\mathbf{u}) = (w_1(\mathbf{u}), \dots, w_J(\mathbf{u}))$, which is parametrized by some learnable parameters $\boldsymbol{\Theta}$. The patch operator can therefore be written in the following general form

$$D_j(x)f = \sum_{y \in \mathcal{N}(x)} w_j(\mathbf{u}(x,y)) f(y), \quad j = 1, \dots, J, \tag{9}$$

where the summation should be interpreted as an integral in the case we deal with a continuous manifold, and $J$ represents the dimensionality of the extracted patch. A spatial generalization of the convolution on non-Euclidean domains is then given by a template-matching procedure of the form

$$(f \star g)(x) = \sum_{j=1}^{J} g_j \, D_j(x) f. \tag{10}$$

The two key choices in our construction are the pseudo-coordinates $\mathbf{u}$ and the weight functions $\mathbf{w}(\mathbf{u})$. Table 3 shows that other deep learning methods (including the classical CNN on Euclidean domains, DCN and DCNN on graphs, and GCNN and ACNN on manifolds) can be obtained as particular settings of our framework with appropriate definition of $\mathbf{u}$ and $\mathbf{w}(\mathbf{u})$. For example, GCNN and ACNN boil down to using Gaussian kernels on local polar geodesic coordinates $\rho, \theta$ on a manifold, and GCN can be interpreted as applying a triangular kernel on pseudo-coordinates given by the degree of the graph vertices.

In this paper, rather than using fixed handcrafted weight functions we consider parametric kernels with learnable parameters. In particular, a convenient choice is

$$w_j(\mathbf{u}) = \exp(-\tfrac{1}{2}(\mathbf{u} - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1} (\mathbf{u} - \boldsymbol{\mu}_j)), \tag{11}$$

where $\boldsymbol{\Sigma}_j$ and $\boldsymbol{\mu}_j$ are learnable $d \times d$ and $d \times 1$ covariance matrix and mean vector of a Gaussian kernel, respectively. Formulae (9–10) can thus be interpreted as a gaussian mixture model (GMM). We further restrict the covariances to have diagonal form, resulting in $2d$ parameters per kernel, and a total of $2Jd$ parameters for the patch operator.

While extremely simple, we show in the next section that these additional degrees of freedom afford our architecture sufficient complexity allowing it to outperform existing approaches. More complex versions of the weighting functions could include additional non-linear transformation of

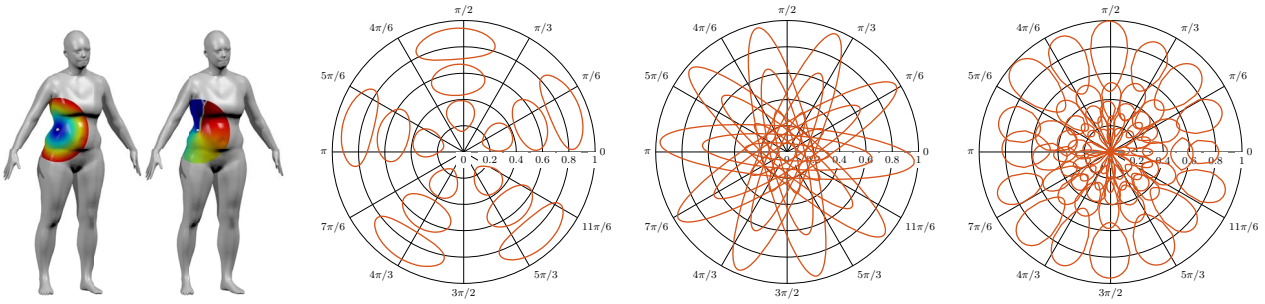


Figure 1. Left: intrinsic local polar coordinates $\rho, \theta$ on manifold around a point marked in white. Right: patch operator weighting functions $w_i(\rho, \theta)$ used in different generalizations of convolution on the manifold (hand-crafted in GCNN and ACNN and learned in MoNet). All kernels are $L_\infty$-normalized; red curves represent the $0.5$ level set.

the pseudo-coordinates $\mathbf{u}$ before feeding them to the Gaussian kernel, or even more general network-in-a-network architectures [25].

## 5. Results

### 5.1. Images

In our first experiment, we applied the proposed method on a classical task of handwritten digit classification in the MNIST dataset [23]. While almost trivial by todays standards, we nevertheless use this example to visualize an important advantage of our approach over spectral graph CNN methods. Our experimental setup followed [13]. The $28 \times 28$ images were represented as graphs, where vertices correspond to (super)pixels and edges represent their spatial relations. We considered two constructions: all images represented on the *same graph* (regular grid) and each image represented as a *different graph* (Figure 2 left and right, respectively). Furthermore, we varied the graph size: the full and $\frac{1}{4}$ grids contained $728$ and $196$ vertices, respectively, while the superpixel-based graphs contained $300$, $150$, and $75$ vertices.

Three methods were compared: classical CNN LeNet5 architecture [23] (containing two convolutional, two max-pooling, and one fully-connected layer, applied on regular grids only), spectral ChebNet[13] and the proposed MoNet. We used a standard splitting of the MNIST dataset into training-, testing-, and validation sets of sizes 55K, 10K, and 5K images, respectively. LeNet used $2\times2$ max-pooling; in ChebNet and MoNet we used three convolutional layers, interleaved with pooling layers based on the Graclus method [14] to coarsen the graph by a factor of four.

For MoNet, we used polar coordinates $\mathbf{u} = (\rho, \theta)$ of pixels (respectively, of superpixel barycenters) to produce the patch operator; as the weighting functions of the patch operator, $25$ Gaussian kernels (initialized with random means and variances) were used. Training was done with 350K iterations of Adam method [20], initial learning rate $10^{-4}$, regularization factor $10^{-4}$, dropout probability $0.5$, and batch size of $10$.

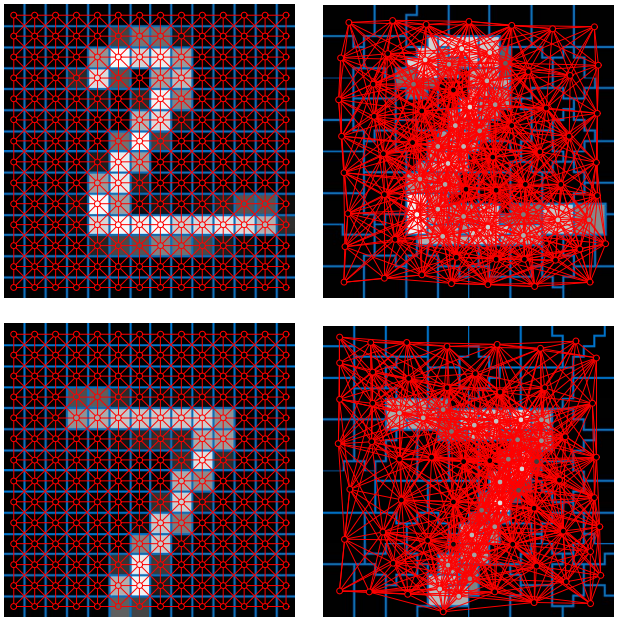


Figure 2. Representation of images as graphs. Left: regular grid (the graph is fixed for all images). Right: graph of superpixel adjacency (different for each image). Vertices are shown as red circles, edges as red lines.

Table 2 summarizes the performance of different algorithms. On regular grids, all the methods perform approximately equally well. However, when applying ChebNet on superpixel-based representations, the performance drops dramatically (by up to almost 25%). In this case, each image is represented as a different graph and the filters used by the ChebNet model fail to generalize well across graphs. The effect is most pronounced on smaller graphs (150 and 75 superpixels) that vary strongly among each other. In contrast, the proposed MoNet approach manifests consistently high accuracy, and only a light performance degradation is observed when the image presentation is too coarse (75 superpixels).

Table 2. Classification accuracy of classical Euclidean CNN (LeNet5), spectral CNN (ChebNet) and the proposed approach (MoNet) on different versions of the MNIST dataset. The setting of all the input images sharing the same graph is marked with *.

| Dataset | LeNet5 [23] | ChebNet [13] | **MoNet** |
|---|---|---|---|
| *Full grid | 99.33% | 99.14% | 99.19% |
| $*\frac{1}{4}$ grid | 98.59% | 97.70% | 98.16% |
| 300 Superpixels | - | 88.05% | **97.30%** |
| 150 Superpixels | - | 80.94% | **96.75%** |
| 75 Superpixels | - | 75.62% | **91.11%** |

## 5.2. Graphs

In the second experiment, we address the problem of vertex classification on generic graphs. We used the popular Cora and PubMed [32] citation graphs as our datasets. In each dataset, a vertex represents a scientific publication (2708 vertices in Cora and 19717 in PubMed, respectively), and an undirected unweighted edge represents a citation (5429 and 44338 edges in Cora and PubMed). For each vertex, a feature vector representing the content of the paper is given (1433-dimensional binary feature vectors in Cora, and 500-dimensional tf-idf weighted word vectors in PubMed). The task is to classify each vertex into one of the groundtruth classes (7 in Cora and 3 in PubMed).

We followed verbatim the experimental settings presented in [44, 21]. The training sets consisted of 20 samples per class; the validation and test sets consisted of 500 and 1000 disjoint vertices. The validation set was chosen in order to reflect the probability distribution of the various classes over the entire dataset. We compared our approach to all the methods compared in [21].

For MoNet, we used the degrees of the nodes as the input pseudo-coordinates $\mathbf{u}(x,y) = (\frac{1}{\sqrt{\deg(x)}}, \frac{1}{\sqrt{\deg(y)}})^\top$; these coordinates underwent an additional transformation in the form of a fully-connected neural network layer $\tilde{\mathbf{u}}(x,y) = \tanh(\mathbf{A}\mathbf{u}(x,y) + \mathbf{b})$, where the $r \times 2$ matrix $\mathbf{A}$ and $r \times 1$ vector $\mathbf{b}$ were also learned (we used $r = 2$ for Cora and $r = 3$ for PubMed). The Gaussian kernels were applied on coordinates $\tilde{\mathbf{u}}(x,y)$ yielding patch operators of the form

$$D_j(x) f_l = \sum_{y \in \mathcal{N}(x)} e^{-\frac{1}{2}(\tilde{\mathbf{u}}(x,y)-\boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\tilde{\mathbf{u}}(x,y)-\boldsymbol{\mu}_j)} f_l(y),$$

where $\boldsymbol{\Sigma}_j$, $\boldsymbol{\mu}_j$, $j = 1, \dots, J$ are the $r \times r$ and $r \times 1$ covariance matrices and mean vectors of the Gaussian kernels, respectively. DCNN, GCN and MoNet were trained in the same way in order to give a fair comparison (see training details in Table 3). The $L_2$-regularization weights for MoNet were $\gamma = 10^{-2}$ and $5 \times 10^{-2}$ for Cora and PubMed, respectively; for DCNN and GCN we used the values suggested by the authors in [3] and [21].

Table 3. Learning configuration used for Cora and PubMed experiments.

| | Cora | PubMed |
|---|---|---|
| Learning Algorithm | Adam | Adam |
| Number of epochs | 3000 | 1000 |
| Validation frequency | 0.01 | 0.04 |
| Learning rate | 0.1 | 0.1 |
| Decay rate | $10^{-1}$ | - |
| Decay epochs | 1500, 2500 | - |
| Early stopping | No | No |

Table 4. Vertex classification accuracy on the Cora and PubMed datasets following the splitting suggested in [44]. Learning methods (DCNN, GCNN and MoNet) were trained and tested fifty times for showing their average behavior with different initializations.

| Method | Cora | PubMed |
|---|---|---|
| ManiReg [4] | 59.5% | 70.7% |
| SemiEmb [42] | 59.0% | 71.1% |
| LP [45] | 68.0% | 63.0% |
| DeepWalk [28] | 67.2% | 65.3% |
| Planetoid [44] | 75.7% | 77.2% |
| DCNN [3] | $76.80 \pm 0.60\%$ | $73.00 \pm 0.52\%$ |
| GCN [21] | $81.59 \pm 0.42\%$ | $78.72 \pm 0.25\%$ |
| **MoNet** | $\mathbf{81.69} \pm 0.48\%$ | $\mathbf{78.81} \pm 0.44\%$ |

The vertex classification results of different methods are summarized in Table 4. MoNet compares favorably to other approaches. The tuning of the network hyper-parameters has been fundamental in this case for avoiding overfitting, due to a very small size of the training set. Being more general, our architecture is more complex compared to GCN and DCNN and requires an appropriate regularization to be used in such settings. At the same time, the greater complexity of our framework might prove advantageous when applied to larger and more complex data.

## 5.3. Manifolds

The last application we consider is learning dense intrinsic correspondence between collections of 3D shapes represented as discrete manifolds. For this purpose, correspondence is cast as a labelling problem, where one tries to label each vertex of a given query shape $\mathcal{X}$ with the index of a corresponding point on some reference shape $\mathcal{Y}$ [30, 26, 7]. Let $n$ and $m$ denote the number of vertices in $\mathcal{X}$ and $\mathcal{Y}$, respectively. For a point $x$ on a query shape, the last layer of the network is soft-max, producing an $m$-dimensional output $\mathbf{f}(x)$ that is interpreted as a probability distribution on $\mathcal{Y}$ (the probability of $x$ mapped to $y$). Learning is done by minimizing the standard logistic regression cost [7].

**Meshes.** We reproduced verbatim the experiments of

[26, 7] on the FAUST humans dataset [5], comparing to the methods reported therein. The dataset consisted of 100 watertight meshes representing 10 different poses for 10 different subjects with exact ground-truth correspondence. Each shape was represented as a mesh with 6890 vertices; the first subject in first pose was used as the reference. For all the shapes, point-wise 544-dimensional SHOT descriptors (local histogram of normal vectors) [39] were used as input data. We used MoNet architecture with 3 convolutional layers, replicating the architectures of [26, 7]. First 80 subjects in all the poses were used for training; the remaining 20 subjects were used for testing. The output of the network was refined using the intrinsic product manifold filter [40] in order to remove some local outliers.

Correspondence quality was evaluated using the Princeton benchmark [19], plotting the percentage of matches that are at most $r$-geodesically distant from the ground-truth correspondence on the reference shape. For comparison, we report the performance of blended maps [19], random forests [30], GCNN [26], ADD [8], and ACNN [7].

Figure 1 shows the weighting functions of the patch operator that are fixed in GCNN and ACNN architectures, and part of the learnable parameters in the proposed MoNet. The patch operators of GCNN and ACNN can be obtained as a particular configuration of MoNet, implying that if trained correctly, the new model can only improve w.r.t. the previous ones. Figure 3 depicts the evaluation results, showing that MoNet significantly outperforms the competing approaches. In particular, close to 90% of points have *zero* error, and for 99% of the points the error is below 4cm. Figure 5 shows the point-wise geodesic correspondence error of our method, and Figure 6 visualizes the obtained correspondence using texture transfer.

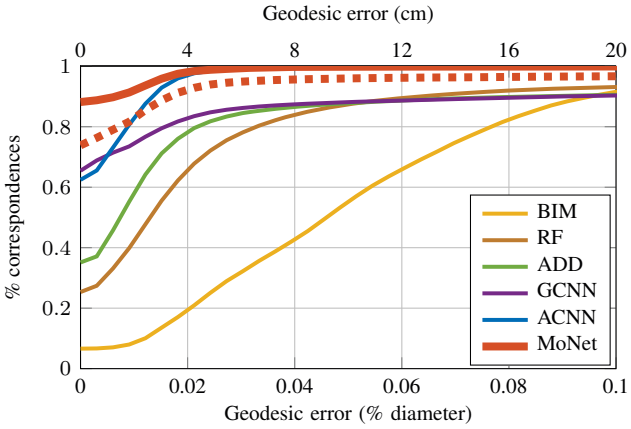


Figure 3. Shape correspondence quality obtained by different methods on the FAUST humans dataset. The raw performance of MoNet is shown in dotted curve.

**Range maps.** Finally, we repeated the shape correspondence experiment on range maps synthetically generated from FAUST meshes. For each subject and pose, we produced 10 rangemaps in $100\times180$ resolution, covering shape rotations around the $z$-axis with increments of 36 degrees (total of 1000 range maps), keeping the ground-truth correspondence. We used MoNet architecture with 3 convolutional layers and local SHOT descriptors as input data. Training and testing set splitting was done as previously.

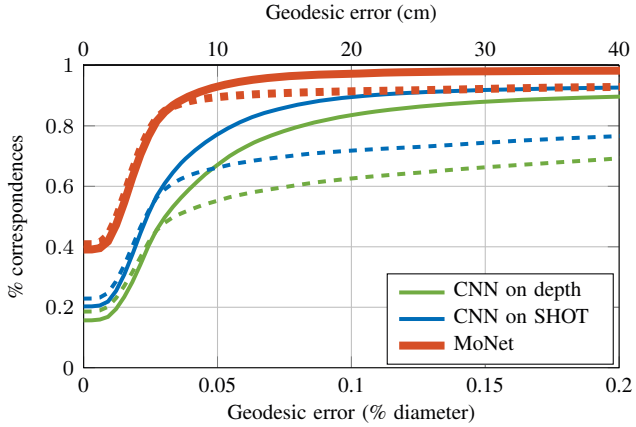


Figure 4. Shape correspondence quality obtained by different methods on FAUST range maps. For comparison, we show the performance of a Euclidean CNN with a comparable 3-layer architecture. The raw performance is shown as dotted curve.

Figure 4 shows the quality of correspondence computed using the Princeton protocol. For comparison, we show the performance of a standard Euclidean CNN in equivalent architecture (3 convolutional layers) applied on raw depth values and on SHOT descriptors. Our approach clearly shows a superior performance. Figure 7 shows the point-wise geodesic correspondence error. Figure 8 shows a qualitative visualization of correspondence using similar color code for corresponding vertices. We also show correspondence on shapes from SCAPE [2] and TOSCA [9] datasets.

## 6. Conclusions

We proposed a spatial-domain model for deep learning on non-Euclidean domains such as manifolds and graphs. Our approach generalizes several previous techniques that can be obtained as particular instances thereof. Extensive experimental results show that our model is applicable to different geometric deep learning tasks, achieving state-of-the-art results. In deformable 3D shape analysis applications, the key advantage of our approach is that it is intrinsic and thus deformation-invariant *by construction*, as opposed to Euclidean models [36, 43, 41, 29] that in general require significantly higher complexity and huge training sets to *learn* the deformation invariance. We believe that our framework will be of broad interest to the CVPR community and will publish all code and data.

**Acknowledgments.** This work was supported in part by the ERC grants Nos. 307047 (COMET) and 724228 (LEMAN), Google Faculty Research Award, and Nvidia equipment grant.

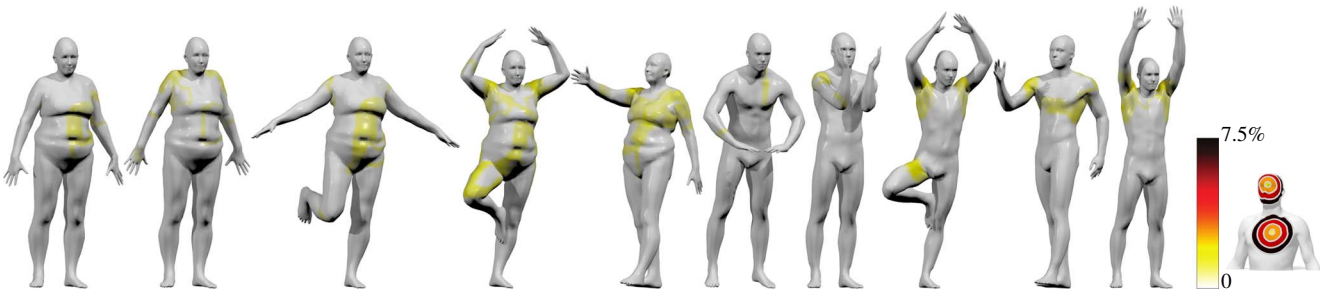


Figure 5. Pointwise error (geodesic distance from groundtruth) of MoNet on the FAUST humans dataset. For visualization clarity, the error values are saturated at 7.5% of the geodesic diameter, which corresponds to approximately 15 cm. Hot colors represent large errors.

Figure 6. Examples of correspondence on the FAUST humans dataset obtained by the proposed MoNet method. Shown is the texture transferred from the leftmost reference shape to different subjects in different poses by means of our correspondence.

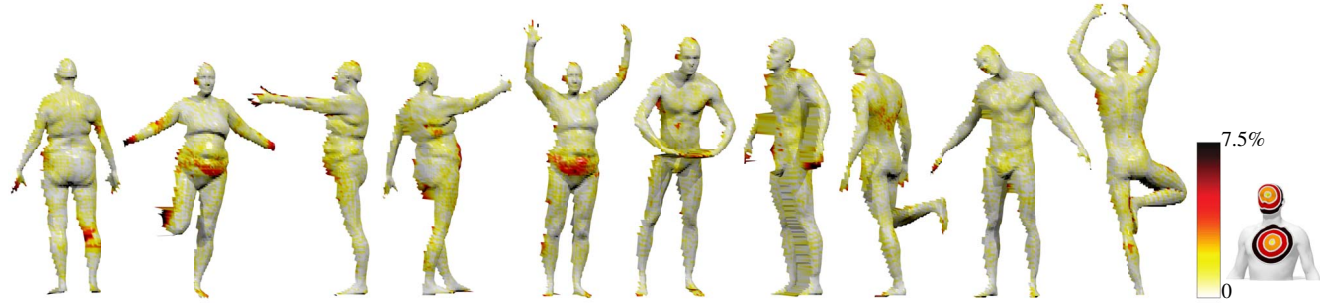


Figure 7. Pointwise error (geodesic distance from groundtruth) of MoNet on FAUST range maps. For visualization clarity, the error values are saturated at 7.5% of the geodesic diameter, which corresponds to approximately 15 cm. Hot colors represent large errors.

Figure 8. Visualization of correspondence on FAUST range maps as color code (corresponding points are shown in the same color). Full reference shape is shown on the left. Bottom row show examples of additional shapes from SCAPE and TOSCA datasets.

## References


[1] M. Andreux, E. Rodolà, M. Aubry, and D. Cremers. Anisotropic Laplace-Beltrami operators for shape analysis. In *Proc. NORDIA*, 2014.

[2] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. In *TOG*, volume 24, pages 408–416, 2005.

[3] J. Atwood and D. Towsley. Diffusion-convolutional neural networks. *arXiv:1511.02136v2*, 2016.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.

[5] F. Bogo, J. Romero, M. Loper, and M. J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proc. CVPR*, 2014.

[6] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. *Computer Graphics Forum*, 34(5):13–23, 2015.

[7] D. Boscaini, J. Masci, E. Rodolà, and M. M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Proc. NIPS*, 2016.

[8] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers. Anisotropic diffusion descriptors. *Computer Graphics Forum*, 35(2):431–441, 2016.

[9] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical geometry of non-rigid shapes*. Springer, 2008.

[10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond Euclidean data. *arXiv:1611.08097*, 2016.

[11] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *Proc. ICLR*, 2013.

[12] S. Cao, W. Lu, and Q. Xu. GraRep: Learning graph representations with global structural information. In *Proc. IKM*, 2015.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. NIPS*, 2016.

[14] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: a multilevel approach. *PAMI*, 29(11):1944–1957, 2007.

[15] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proc. NIPS*, 2015.

[16] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proc. IJCNN*, 2005.

[17] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proc. KDD*, 2016.

[18] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv:1506.05163*, 2015.

[19] V. Kim, Y. Lipman, and T. Funkhouser. Blended intrinsic maps. *ACM Trans. Graphics*, 30(4):79, 2011.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[21] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016.

[22] I. Kokkinos, M. Bronstein, R. Litman, and A. Bronstein. Intrinsic shape context descriptors for deformable shapes. In *Proc. CVPR*, 2012.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[24] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv:1511.05493*, 2015.

[25] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[26] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *Proc. 3DRR*, 2015.

[27] T. Mikolov and J. Dean. Distributed representations of words and phrases and their compositionality. *Proc. NIPS*, 2013.

[28] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online learning of social representations. In *Proc. KDD*, 2014.

[29] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *Proc. CVPR*, 2016.

[30] E. Rodolà, S. Rota Bulò, T. Windheuser, M. Vestner, and D. Cremers. Dense non-rigid shape correspondence using random forests. In *Proc. CVPR*, 2014.

[31] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[32] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[33] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Sig. Proc. Magazine*, 30(3):83–98, 2013.

[34] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *App. and Comp. Harmonic Analysis*, 40(2):260–291, 2016.

[35] A. Sinha, J. Bai, and K. Ramani. Deep learning 3D shape surfaces using geometry images. In *Proc. ECCV*, 2016.

[36] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proc. ICCV*, 2015.

[37] S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. *arXiv:1605.07736*, 2016.

[38] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale information network embedding. In *Proc. WWW*, 2015.

[39] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Proc. ECCV*, 2010.

[40] M. Vestner, R. Litman, E. Rodolà, A. M. Bronstein, and D. Cremers. Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In *Proc. CVPR*, 2017.

[41] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *Proc. CVPR*, 2016.

[42] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. 2012.

[43] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D shapenets: A deep representation for volumetric shapes. In *Proc. CVPR*, 2015.

[44] Z. Yang, W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv:1603.08861*, 2016.

[45] X. Zhu, Z. Ghahramani, J. Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. ICML*, 2003.