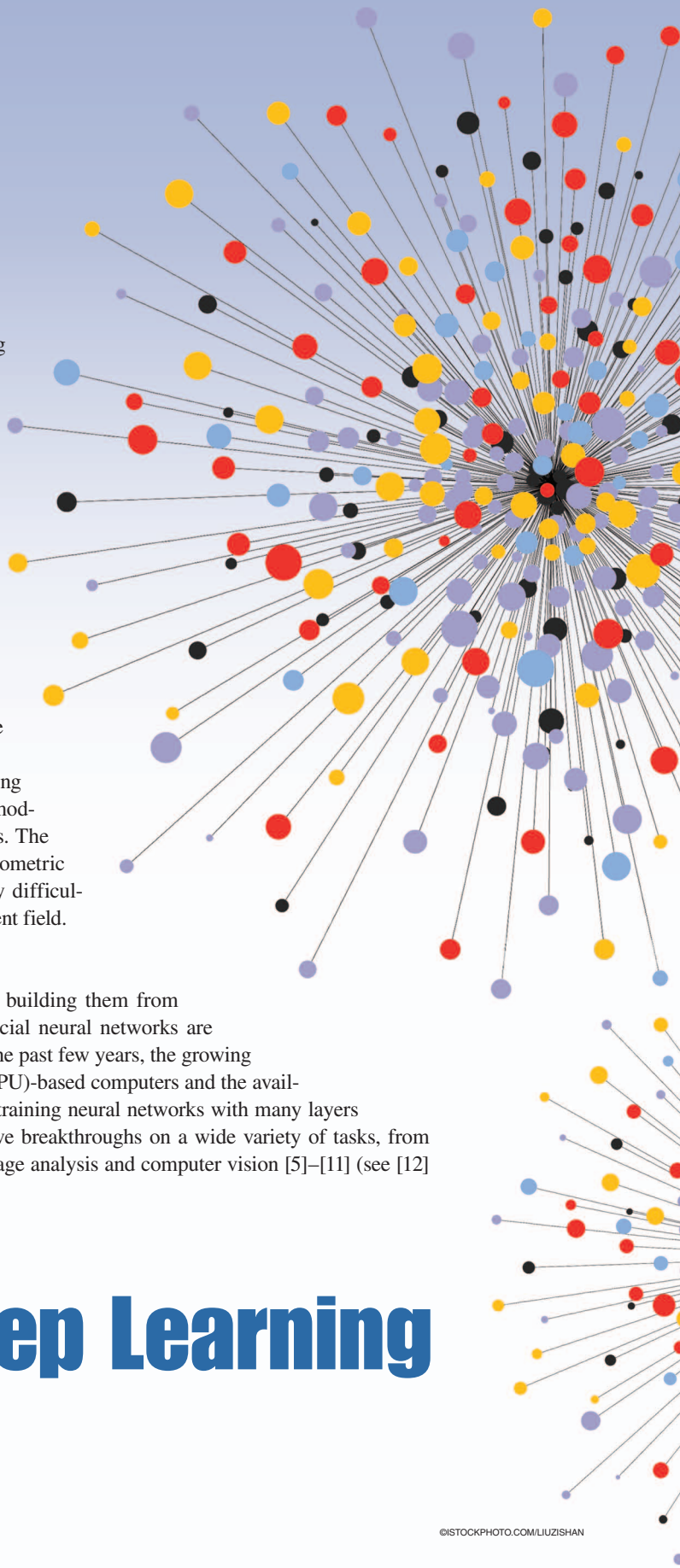
Michael M. Bronstein, Joan Bruna, Yann LeCun,
Arthur Szlam, and Pierre Vandergheynst



# Geometric Deep Learning

## *Going beyond Euclidean data*

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

*Geometric deep learning* is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

### Overview of deep learning

*Deep learning* refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

*Digital Object Identifier 10.1109/MSP.2017.2693418*
*Date of publication: 11 July 2017*

and [13] for many additional examples of successful applications of deep learning). Today, deep learning has matured into a technology that is widely used in commercial applications, including Siri speech recognition in Apple iPhone, Google text translation, and Mobileye vision-based technology for autonomously driving cars.

One of the key reasons for the success of deep neural networks is their ability to leverage statistical properties of the data, such as stationarity and compositionality through local statistics, which are present in natural images, video, and speech [14], [15]. These statistical properties have been related to physics [16] and formalized in specific classes of convolutional neural networks (CNNs) [17]–[19]. In image analysis applications, one can consider images as functions on the Euclidean space (plane), sampled on a grid. In this setting, stationarity is owed to shift invariance, locality is due to the local connectivity, and compositionality stems from the multiresolution structure of the grid. These properties are exploited by convolutional architectures [20], which are built of alternating convolutional and downsampling (pooling) layers. The use of convolutions has a twofold effect. First, it allows extracting local features that are shared across the image domain and greatly reduces the number of parameters in the network with respect to generic deep architectures (and thus also the risk of overfitting), without sacrificing the expressive capacity of the network. Second, the convolutional architecture itself imposes some priors about the data, which appear very suitable especially for natural images [17]–[19], [21].

While deep-learning models have been particularly successful when dealing with speech, image, and video signals, in which there are an underlying Euclidean structure, recently there has been a growing interest in trying to apply learning on non-Euclidean geometric data. Such kinds of data arise in numerous applications. For instance, in social networks, the characteristics of users can be modeled as signals on the vertices of the social graph [22]. Sensor networks are graph models of distributed interconnected sensors, whose readings are modeled as time-dependent signals on the vertices. In genetics, gene expression data are modeled as signals defined on the regulatory network [23]. In neuroscience, graph models are used to represent anatomical and functional structures of the brain. In computer graphics and vision, three-dimensional (3-D) objects are modeled as Riemannian manifolds (surfaces) endowed with properties such as color texture.

The non-Euclidean nature of such data implies that there are no such familiar properties as global parameterization, common system of coordinates, vector space structure, or shift invariance. Consequently, basic operations like convolution that are taken for granted in the Euclidean case are even not well defined on non-Euclidean domains. The purpose of this article is to show different methods of translating the key ingredients of successful deep-learning methods, such as CNNs, to non-Euclidean data.

## Geometric learning problems

Broadly speaking, we can distinguish between two classes of geometric learning problems. In the first class of problems, the goal is to characterize the structure of the data. The second class of problems deals with analyzing functions defined on a given non-Euclidean domain. These two classes are related, because understanding the properties of functions defined on a domain conveys certain information about the domain, and vice versa, the structure of the domain imposes certain properties on the functions on it.

### *Structure of the domain*

As an example of the first class of problems, assume to be given a set of data points with some underlying low-dimensional structure embedded into a high-dimensional Euclidean space. Recovering that low-dimensional structure is often referred to as *manifold learning* or *nonlinear dimensionality reduction* and is an instance of unsupervised learning (note that the notion of manifold in this setting can be considerably more general than a classical smooth manifold; see, e.g.,

[24] and [25]). Many methods for nonlinear dimensionality reduction consist of two steps: first, they start with constructing a representation of local affinity of the data points (typically, a sparsely connected graph). Second, the data points are embedded into a low-dimensional space, trying to preserve some criterion of the original affinity. For example, spectral embeddings tend to map points with many connections between them to nearby locations, and multidimensional scaling (MDS)-type methods try to preserve global information, such as graph geodesic distances. Examples of manifold learning include different flavors of MDS [26], locally linear embedding [27], stochastic neighbor embedding [28], spectral embeddings, such as Laplacian eigenmaps [29] and diffusion maps [30], and deep models [31]. Instead of embedding the vertices, the graph structure can be processed by decomposing it into small subgraphs called *motifs* [36] or *graphlets* [37]. Finally, most recent approaches [32]–[34] tried to apply the successful word-embedding model [35] to graphs.

**Today, deep learning has matured into a technology that is widely used in commercial applications.**

In some cases, the data are presented as a manifold or graph at the outset, and the first step of constructing the affinity structure described previously is unnecessary. For instance, in computer graphics and vision applications, one can analyze 3-D shapes represented as meshes by constructing local geometric descriptors capturing, e.g., curvature-like properties [38], [39]. In social network analysis applications the topological structure of the social graph representing the social relations between people carries important insights allowing, e.g., to classify the vertices and detect communities [40]. In natural-language processing, words in a corpus can be represented by the co-occurrence graph, where two words are connected if they often appear near each other [41].

### Data on a domain

Our second class of problems deals with analyzing functions defined on a given non-Euclidean domain. We can further break down such problems into two subclasses: problems where the domain is fixed and those where multiple domains are given. For example, assume that we are given the geographic coordinates of the users of a social network, represented as a time-dependent signal on the vertices of the social graph. An important application in location-based social networks is to predict the position of the user given his or her past behavior as well as that of his or her friends [42]. In this problem, the domain (social graph) is assumed to be fixed; methods of signal processing on graphs, which have previously been reviewed in *IEEE Signal Processing Magazine* [43], can be applied to this setting, in particular, to define an operation similar to convolution in the spectral domain. This, in turn, allows generalizing CNN models to graphs [44], [45]. In computer graphics and vision applications, finding similarity and correspondence between shapes are examples of the second subclass of problems: each shape is modeled as a manifold, and one has to work with multiple such domains. In this setting, a generalization of convolution in the spatial domain using local charting [46]–[48] appears to be more appropriate.

### Brief history

The main focus of this review is on this second class of problems, namely, learning functions on non-Euclidean structured domains, and, in particular, attempts to generalize the popular CNNs to such settings. The first attempts to generalize neural networks to graphs we are aware of are due to Gori et al. [49], who proposed a scheme combining recurrent neural networks (RNNs) and random walk models. This approach went almost unnoticed, reemerging in a modern form in [50] and [51] due to the renewed recent interest in deep learning. The first formulation of CNNs on graphs is due to Bruna et al. [52], who used the definition of convolutions in the spectral domain. Their article, while being of conceptual importance, came with significant computational drawbacks that fell short of a truly useful method. These drawbacks were subsequently addressed in the follow-up works of Henaff et al. [44] and Defferrard et al. [45]. In the latter article, graph CNNs (GCNNs) allowed achieving some state-of-the-art results.

In a parallel effort in the computer vision and graphics community, Masci et al. [47] showed the first CNN model on meshed surfaces, resorting to a spatial definition of the convolution operation based on local intrinsic patches. Among other applications, such models were shown to achieve state-of-the-art performance in finding correspondence between deformable 3-D shapes. Follow-up works proposed different construction of intrinsic patches on point clouds [48], [53] and general graphs [54].

The interest in deep learning on graphs or manifolds has exploded in the past year, resulting in numerous attempts to apply these methods to a broad spectrum of problems ranging from biochemistry [55] to recommender systems [56]. Because such applications originate in different fields that usually do not cross-fertilize, publications in this domain tend to use different terminology and notation, making it difficult for a newcomer to grasp the foundations and current state-of-the-art methods. We believe that our article comes at the right time, attempting to systemize and bring some order into the field.

### Signal processing, differential geometry, and graph theory

Geometric deep-learning frameworks dealt with in this paper are based on notions in differential geometry and graph theory. Unfortunately, these topics are insufficiently known in the signal processing community, and to our knowledge, there is no introductory-level reference treating these so different structures in a common way. One of our goals is to provide an accessible overview of these models, resorting as much as possible to the intuition of traditional signal processing.

One of the key differences between Euclidean and non-Euclidean learning settings is the lack of traditional operations such as convolutions. Various non-Euclidean convolutional architectures differ in the way a convolution-like operation is

formulated on graphs and manifolds. One way is to resort to the analogy of the convolution theorem, defining the convolution in the spectral domain. An alternative is to think of the convolution as a template matching in the spatial domain. Such a distinction is, however, far from being clear-cut: as we will see, some approaches draw their formulation from the spectral domain, essentially boiling down to applying filters in the spatial domain. It is also possible to combine these two approaches, resorting to spatio-frequency analysis techniques, such as wavelets or the windowed Fourier transform. We have provided sidebars to illustrate important concepts, and Table 1 lists the notations used throughout the article. Additional materials, data, and examples of code are available at geometricdeeplearning.com. Table 2 provides a summary of the geometric deep-learning methods presented in this article.

**Table 1. The notations used in this article.**

| Notation | |
|---|---|
| $\mathbb{R}^m$ | $m$-dimensional Euclidean space |
| $a, \mathbf{a}, \mathbf{A}$ | Scalar, vector, matrix |
| $\bar{a}$ | Complex conjugate of $a$ |
| $\Omega, x$ | Arbitrary domain, coordinate on it |
| $f \in L^2(\Omega)$ | Square-integrable function on $\Omega$ |
| $\delta_{x'}(x), \delta_{ij}$ | Delta function at $x'$, Kronecker delta |
| $\{f_i, y_i\}_{i \in I}$ | Training set |
| $\mathcal{T}_v$ | Translation operator |
| $\tau, \mathcal{L}_\tau$ | Deformation field, operator |
| $\hat{f}$ | Fourier transform of $f$ |
| $f \star g$ | Convolution of $f$ and $g$ |
| $\mathcal{X}, T\mathcal{X}, T_x\mathcal{X}$ | Manifold, its tangent bundle, tangent space at $x$ |
| $\langle \cdot, \cdot, \rangle_{T\mathcal{X}}$ | Riemannian metric |
| $f \in L^2(\mathcal{X})$ | Scalar field on manifold $\mathcal{X}$ |
| $F \in L^2(T\mathcal{X})$ | Tangent vector field on manifold $\mathcal{X}$ |
| $A^*$ | Adjoint of operator $A$ |
| $\nabla, div, \Delta$ | Gradient, divergence, Laplace operators |
| $\mathcal{V}, \mathcal{E}, \mathcal{F}$ | Vertices and edges of a graph, faces of a mesh |
| $w_{ij}, \mathbf{W}$ | Weight matrix of a graph |
| $f \in L^2(\mathcal{V})$ | Functions on vertices of a graph |
| $F \in L^2(\mathcal{E})$ | Functions on edges of a graph |
| $\phi_i, \lambda_i$ | Laplacian eigenfunctions, eigenvalues |
| $h_t(\cdot, \cdot)$ | Heat kernel |
| $\mathbf{\Phi}_k$ | Matrix of first $k$ Laplacian eigenvectors |
| $\mathbf{\Lambda}_k$ | Diagonal matrix of first $k$ Laplacian eigenvalues |
| $\xi$ | Pointwise nonlinearity (ReLU) |
| $\gamma_{l,l'}(x), \Gamma_{l,l'}$ | Convolutional filter in spatial and spectral domain |

## Deep learning on Euclidean domains

### *Geometric priors*

Consider a compact $d$-dimensional Euclidean domain $\Omega = [0,1]^d \subset \mathbb{R}^d$ on which square-integrable functions $f \in L^2(\Omega)$ are defined (e.g., in image analysis applications, images can be thought of as functions on the unit square $\Omega = [0,1]^2$). We consider a generic supervised learning setting, in which an unknown function $y : L^2(\Omega) \to \mathcal{Y}$ is observed on a training set

$$\{f_i \in L^2(\Omega), y_i = y(f_i)\}_{i \in I}. \tag{1}$$

In a supervised classification setting, the target space $\mathcal{Y}$ can be thought discrete, with $|\mathcal{Y}|$ being the number of classes. In a multiple object recognition setting, we can replace $\mathcal{Y}$ by a multi-$K$-dimensional simplex, which represents the posterior class probabilities $p(y|x)$. In regression tasks, we may consider $\mathcal{Y} = \mathbb{R}^m$. In the vast majority of computer-vision and speech-analysis tasks, there are several crucial prior assumptions on the unknown function $y$. As we will see in the following sections, these assumptions are effectively exploited by CNN architectures.

#### Stationarity

Let

$$\mathcal{T}_v f(x) = f(x - v), \qquad x, v \in \Omega, \tag{2}$$

be a translation operator acting on functions $f \in L^2(\Omega)$ [we assume periodic boundary conditions to ensure that the operation is well defined over $L^2(\Omega)$]. Our first assumption is that the function $y$ is either invariant or equivariant with respect to translations, depending on the task. In the former case, we have $y(\mathcal{T}_v f) = y(f)$ for any $f \in L^2(\Omega)$ and $v \in \Omega$. This is typically the case in object classification tasks. In the latter, we have $y(\mathcal{T}_v f) = \mathcal{T}_v y(f)$, which is well defined when the output of the model is a space in which translations can act (e.g., in problems of object localization, semantic segmentation, or motion estimation). Our definition of invariance should not be confused with the traditional notion of translation invariant systems in signal processing, which corresponds to translation equivariance in our language (because the output translates whenever the input translates).

#### Local deformations and scale separation

Similarly, a deformation $\mathcal{L}_\tau$, where $\tau : \Omega \to \Omega$ is a smooth vector field, acts on $L^2(\Omega)$ as $\mathcal{L}_\tau f(x) = f(x - \tau(x))$. Deformations can model local translations, changes in point of view, rotations, and frequency transpositions [18]. Most tasks studied in computer vision are not only translation invariant/equivariant but also stable with respect to local deformations [57], [18]. In tasks that are translation invariant, we have

$$|y(\mathcal{L}_\tau f) - y(f)| \approx \|\nabla \tau\|, \tag{3}$$

## CNN Architecture

CNNs are currently among the most successful deep-learning architectures in a variety of tasks; in particular, in computer vision. A typical CNN used in computer-vision applications (see Figure S1) consists of multiple convolutional layers (6), passing the input image through a set of filters $\Gamma$ followed by pointwise nonlinearity $\xi$ (typically, half-rectifiers $\xi(z) = \max(0, z)$ are used, although practitioners have experimented with a diverse range of choices [13]). The model can also include a bias term, which is equivalent to adding a constant coordinate to the input.

A network composed of $K$ convolutional layers put together $U(f) = (C_{\Gamma^{(K)}} \ldots \circ C_{\Gamma^{(2)}} \circ C_{\Gamma^{(1)}})(f)$ produces pixel-wise features that are covariant with respect to translation and approximately covariant to local deformations. Typical computer-vision applications requiring covariance are semantic image segmentation [8] or motion estimation [59].

In applications requiring invariance, such as image classification [7], the convolutional layers are typically interleaved with pooling layers (8) progressively reducing the resolution of the image passing through the network. Alternatively, one can integrate the convolution and downsampling in a single linear operator (convolution with stride). Recently, some authors have also experimented with convolutional layers that increase the spatial resolution using interpolation kernels [60]. These kernels can be learned efficiently by mimicking the so-called *algorithme à trous* [61], also referred to as *dilated convolution*.

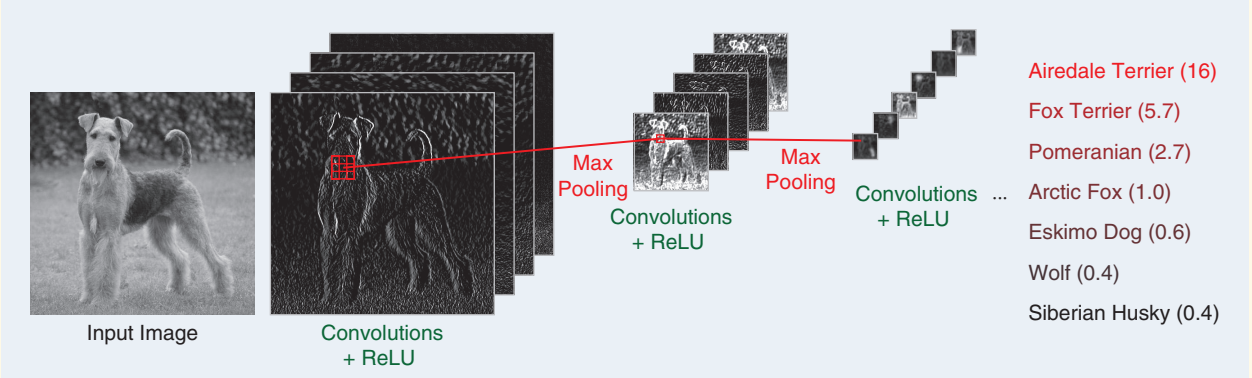



**FIGURE S1.** The typical CNN architecture used in computer-vision applications such as image classification.

**Table 2. The dichotomy of geometric deep-learning methods.**

| Method | Type | Data |
|---|---|---|
| SCNN [52] | Spectral | Graph |
| GCNN/ChebNet [45] | Spectrum free | Graph |
| GCN [77] | Spectrum free | Graph |
| GNN [78] | Spectrum free | Graph |
| Geodesic CNN [47] | Charting | Mesh |
| Anisotropic CNN [48] | Charting | Mesh/point cloud |
| MoNet [54] | Charting | Graph/mesh/point cloud |
| Localized SCNN [89] | Combined | Mesh/point cloud |

for all $f, \tau$. Here, $\|\nabla\tau\|$ measures the smoothness of a given deformation field. In other words, the quantity to be predicted does not change much if the input image is slightly deformed. In tasks that are translation equivariant, we have

$$|y(\mathcal{L}_\tau f) - \mathcal{L}_\tau y(f)| \approx \|\nabla\tau\|. \tag{4}$$

This property is much stronger than the previous one, because the space of local deformations has a high dimensionality, as opposed to the $d$-dimensional translation group. It follows from (3) that we can extract sufficient statistics at a lower spatial resolution by downsampling demodulated localized filter responses without losing approximation power. An important consequence of this is that long-range dependencies can be broken into multiscale local interaction terms, leading to hierarchical models in which spatial resolution is progressively reduced. To illustrate this principle, denote by

$$Y(x_1, x_2; v) = \text{Prob}(f(u) = x_1 \text{ and } f(u + v) = x_2) \tag{5}$$

the joint distribution of two image pixels at an offset $v$ from each other. In the presence of long-range dependencies, this joint distribution will not be separable for any $v$. However, the deformation stability prior states that $Y(x_1, x_2; v) \approx Y(x_1, x_2; v(1 + \epsilon))$ for small $\epsilon$. In other words,

whereas long-range dependencies indeed exist in natural images and are critical to object recognition, they can be captured and downsampled at different scales. This principle of stability to local deformations has been exploited in the computer-vision community in models other than CNNs, for instance, deformable parts models [58]. In practice, the Euclidean domain $\Omega$ is discretized using a regular grid with $n$ points; the translation and deformation operators are still well defined so the above properties also hold in the discrete setting.

### CNNs

Stationarity and stability to local translations are both leveraged in CNNs (see "CNN Architecture" and [1], [12], [13], and references therein for a more in-depth review of CNNs and their applications.) A CNN consists of several convolutional layers of the form $\mathbf{g} = C_\Gamma(\mathbf{f})$, acting on a $p$-dimensional input $\mathbf{f}(x) = (f_1(x), \ldots, f_p(x))$ by applying a bank of filters $\Gamma = (\gamma_{l,l'}), l = 1, \ldots, q, l' = 1, \ldots, p$ and pointwise nonlinearity $\xi$,

$$g_l(x) = \xi\left(\sum_{l'=1}^{p} (f_{l'} \star \gamma_{l,l'})(x)\right), \tag{6}$$

producing a $q$-dimensional output $\mathbf{g}(x) = (g_1(x), \ldots, g_q(x))$ often referred to as the *feature maps*. Here,

$$(f \star \gamma)(x) = \int_\Omega f(x - x')\gamma(x')dx' \tag{7}$$

denotes the standard convolution. According to the local deformation prior, the filters $\Gamma$ have compact spatial support.

Additionally, a downsampling or pooling layer $\mathbf{g} = P(\mathbf{f})$ may be used, defined as

$$g_l(x) = P(\{f_l(x'): x' \in \mathcal{N}(x)\}), \quad l = 1, \ldots, q, \tag{8}$$

where $\mathcal{N}(x) \subset \Omega$ is a neighborhood around $x$ and $P$ is a permutation-invariant function, such as an $L_p$-norm (in the latter case, the choice of $p = 1$, 2, or $\infty$ results in average, energy, or max pooling).

A convolutional network is constructed by composing several convolutional and optionally pooling layers, obtaining a generic hierarchical representation

$$U_\Theta(f) = (C_{\Gamma^{(K)}} \cdots P \cdots \circ C_{\Gamma^{(2)}} \circ C_{\Gamma^{(1)}})(f), \tag{9}$$

where $\Theta = \{\Gamma^{(1)}, \ldots, \Gamma^{(K)}\}$ is the hypervector of the network parameters (all the filter coefficients). The model is said to be deep if it comprises multiple layers, though this notion is rather vague, and one can find examples of CNNs with as few as a couple and as many as hundreds of layers [11]. The output features enjoy translation invariance/covariance depending on whether spatial resolution is progressively lost by means of pooling or kept fixed. Moreover, if one specifies the convolutional tensors to be complex wavelet decomposition operators and uses complex modulus as pointwise nonlinearities, one can provably obtain stability to local deformations [17]. Although this stability is not rigorously proved for generic compactly supported convolutional tensors, it underpins the empirical success of CNN architectures across a variety of computer-vision applications [1].

In supervised learning tasks, one can obtain the CNN parameters by minimizing a task-specific cost $L$ on the training set $\{f_i, y_i\}_{i \in I}$,

$$\min_\Theta \sum_{i \in I} L(U_\Theta(f_i), y_i), \tag{10}$$

for instance, $L(x, y) = \|x - y\|$. If the model is sufficiently complex and the training set is sufficiently representative, when applying the learned model to previously unseen data, one expects $U(f) \approx y(f)$. Although (10) is a nonconvex optimization problem, stochastic optimization methods offer excellent empirical performance. Understanding the structure of the optimization problems (10) and finding efficient strategies for its solution is an active area of research in deep learning [62]–[66].

A key advantage of CNNs explaining their success in numerous tasks is that the geometric priors on which CNNs are based result in a learning complexity that avoids the curse of dimensionality. Thanks to the stationarity and local deformation priors, the linear operators at each layer have a constant number of parameters, independent of the input size $n$ (number of pixels in an image). Moreover, thanks to the multiscale hierarchical property, the number of layers grows at a rate $O(\log n)$, resulting in a total learning complexity of $O(\log n)$ parameters.

## The geometry of manifolds and graphs

Our main goal is to generalize CNN-type constructions to non-Euclidean domains. In this article, by non-Euclidean domains, we refer to two prototypical structures: manifolds and graphs. While arising in very different fields of mathematics (differential geometry and graph theory, respectively), in our context, these structures share several common characteristics that we will try to emphasize throughout our review.

### Manifolds

Roughly, a manifold is a space that is locally Euclidean. One of the simplest examples is a spherical surface modeling our planet: around a point, it seems to be planar, which has led generations of people to believe in the flatness of the Earth. Formally speaking, a (differentiable) $d$-dimensional manifold $\mathcal{X}$ is a topological space where each point $x$ has a neighborhood that is topologically equivalent (homeomorphic) to a $d$-dimensional Euclidean space, called the *tangent space* and denoted by $T_x\mathcal{X}$ [see Figure 1(a)]. The collection of tangent spaces at all points (more formally, their disjoint union) is referred to as the *tangent bundle* and denoted by $T\mathcal{X}$. On each tangent space, we define an inner product $\langle \cdot, \cdot \rangle_{T_x\mathcal{X}}: T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$, which is additionally assumed to depend smoothly on the position $x$. This inner product is

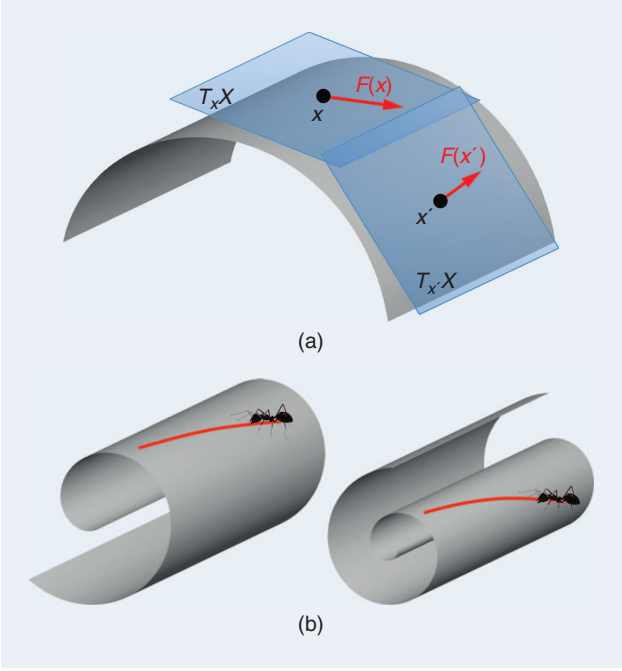



**FIGURE 1.** (a) The tangent space and tangent vectors on a 2-D manifold (surface). (b) Examples of isometric deformations.

called a *Riemannian metric* in differential geometry and allows performing local measurements of angles, distances, and volumes. A manifold equipped with a metric is called a *Riemannian manifold*.

It is important to note that the definition of a Riemannian manifold is completely abstract and does not require a geometric realization in any space. However, a Riemannian manifold can be realized as a subset of a Euclidean space (in which case it is said to be embedded in that space) by using the structure of the Euclidean space to induce a Riemannian metric. The celebrated Nash embedding theorem guarantees that any sufficiently smooth Riemannian manifold can be realized in a Euclidean space of sufficiently high dimension [67]. An embedding is not necessarily unique; two different realizations of a Riemannian metric are called *isometries*.

Two-dimensional (2-D) manifolds (surfaces) embedded into $\mathbb{R}^3$ are used in computer graphics and vision to describe boundary surfaces of 3-D objects, colloquially referred to as *3-D shapes*. This term is somewhat misleading because *3-D* here refers to the dimensionality of the embedding space rather than that of the manifold. Thinking of such a shape as made of infinitely thin material, inelastic deformations that do not stretch or tear it are isometric. Isometries do not affect the metric structure of the manifold, and consequently, they preserve any quantities that can be expressed in terms of the Riemannian metric (called *intrinsic*). Conversely, properties pertaining to the specific realization of the manifold in the Euclidean space are called *extrinsic*. As an intuitive illustration of this difference, imagine an insect that lives on a 2-D surface [Figure 1(b)]. The surface can be placed in the Euclidean space in any way, but as long as it is transformed isometrically, the insect would not notice any difference. The insect in fact does not even know of the existence of the embedding space, as its only world is 2-D. This is an intrinsic viewpoint. A human observer, on the other hand, sees a surface in 3-D space—this is an extrinsic point of view.

## Calculus on manifolds

Our next step is to consider functions defined on manifolds. We are particularly interested in two types of functions: A scalar field is a smooth real function $f:\mathcal{X}\to\mathbb{R}$ on the manifold. A tangent vector field $F:\mathcal{X}\to T\mathcal{X}$ is a mapping attaching a tangent vector $F(x)\in T_x\mathcal{X}$ to each point $x$. As we will see in the following, tangent vector fields are used to formalize the notion of infinitesimal displacements on the manifold. We define the Hilbert spaces of scalar and vector fields on manifolds, denoted by $L^2(\mathcal{X})$ and $L^2(T\mathcal{X})$, respectively, with the following inner products:

$$\langle f,g\rangle_{L^2(\mathcal{X})}=\int_{\mathcal{X}} f(x)\,g(x)\,dx, \tag{11}$$

$$\langle F,G\rangle_{L^2(T\mathcal{X})}=\int_{\mathcal{X}}\langle F(x),G(x)\rangle_{T_x\mathcal{X}}\,dx. \tag{12}$$

Here, $dx$ denotes a $d$-dimensional volume element induced by the Riemannian metric.

In calculus, the notion of derivative describes how the value of a function changes with an infinitesimal change of its argument. One of the big differences distinguishing classical calculus from differential geometry is a lack of vector space structure on the manifold, prohibiting us from naively using expressions like $f(x+dx)$. The conceptual leap that is required to generalize such notions to manifolds is the need to work locally in the tangent space.

To this end, we define the differential of $f$ as an operator $df:T\mathcal{X}\to\mathbb{R}$ acting on tangent vector fields. At each point $x$, the differential can be defined as a linear functional $df(x)=\langle\nabla f(x),\cdot\rangle_{T_x\mathcal{X}}$ acting on tangent vectors $F(x)\in T_x\mathcal{X}$, which model a small displacement around $x$. The change of the function value as the result of this displacement is given by applying the functional to the tangent vector, $df(x)\,F(x)=\langle\nabla f(x),F(x)\rangle_{T_x\mathcal{X}}$, and can be thought of as an extension of the notion of the classical directional derivative.

The operator $\nabla f:L^2(\mathcal{X})\to L^2(T\mathcal{X})$ in the previous definition is called the *intrinsic gradient* and is similar to the classical notion of the gradient defining the direction of the steepest change of the function at a point, with the only difference that the direction is now a tangent vector. Similarly, the intrinsic divergence is an operator $\mathrm{div}:L^2(T\mathcal{X})\to L^2(\mathcal{X})$ acting on tangent vector fields and is (formal) adjoint to the gradient operator [71],

$$\langle F,\nabla f\rangle_{L^2(T\mathcal{X})}=\langle\nabla^* F,f\rangle_{L^2(\mathcal{X})}=\langle-\mathrm{div}F,f\rangle_{L^2(\mathcal{X})}. \tag{13}$$

Physically, a tangent vector field can be thought of as a flow of material on a manifold. The divergence measures the net flow of a field at a point, allowing to distinguish between field sources and sinks. Finally, the Laplacian (or Laplace–Beltrami

## Physical Interpretation of Laplacian Eigenfunctions

Given a function $f$ on the domain $\mathcal{X}$, the Dirichlet energy

$$\varepsilon_{\text{Dir}}(f) = \int_{\mathcal{X}} \| \nabla f(x) \|^2_{T_x\mathcal{X}} \, dx = \int_{\mathcal{X}} f(x) \Delta f(x) \, dx, \tag{S1}$$

measures how smooth it is [the last identity in (S1) stems from (15)]. We are looking for an orthonormal basis on $\mathcal{X}$, containing $k$ smoothest possible functions (Figure S2), by solving the optimization problem

$$\begin{aligned} &\min_{\phi_0} \mathcal{E}_{\text{Dir}}(\phi_0) \ \text{ s.t. } \ \|\phi_0\| = 1 \\ &\min_{\phi_i} \mathcal{E}_{\text{Dir}}(\phi_i) \ \text{ s.t. } \ \|\phi_i\| = 1, i = 1, 2, \ldots k-1 \\ &\qquad\qquad \phi_i \perp \text{span}\{\phi_0, \ \ldots, \phi_{i-1}\}. \end{aligned} \tag{S2}$$

In the discrete setting, when the domain is sampled at $n$ points, (S2) can be rewritten as

$$\min_{\Phi_k \in \mathbb{R}^{n \times k}} \text{trace}(\Phi_k^\top \Delta \Phi_k) \ \text{ s.t. } \ \Phi_k^\top \Phi_k = \mathbf{I}, \tag{S3}$$

where $\Phi_k = (\boldsymbol{\phi}_0, \ldots \boldsymbol{\phi}_{k-1})$. The solution of (S3) is given by the first $k$ eigenvectors of $\Delta$ satisfying

$$\Delta \Phi_k = \Phi_k \Lambda_k, \tag{S4}$$

where $\Lambda_k = \text{diag}(\lambda_0, \ldots, \lambda_{k-1})$ is the diagonal matrix of corresponding eigenvalues. The eigenvalues $0 = \lambda_0 \le \lambda_1 \le \cdots \lambda_{k-1}$ are nonnegative due to the positive semidefiniteness of the Laplacian and can be interpreted as frequencies, where $\phi_0 = \text{const}$ with the corresponding eigenvalue $\lambda_0 = 0$ plays the role of the direct current component.

The Laplacian eigendecomposition can be carried out in two ways. First, (S4) can be rewritten as a generalized eigenproblem $(\mathbf{D} - \mathbf{W})\Phi_k = \mathbf{A}\Phi_k\Lambda_k$, resulting in $\mathbf{A}$-orthogonal eigenvectors, $\Phi_k^\top \mathbf{A} \Phi_k = \mathbf{I}$. Alternatively, introducing a change of variables $\Psi_k = \mathbf{A}^{1/2}\Phi_k$, we can obtain a standard eigendecomposition problem $\mathbf{A}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{A}^{-1/2}\Psi_k = \Psi_k\Lambda_k$ with orthogonal eigenvectors $\Psi_k^\top \Psi_k = \mathbf{I}$. When $\mathbf{A} = \mathbf{D}$ is used, the matrix $\Delta = \mathbf{A}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{A}^{-1/2}$ is referred to as the *normalized symmetric Laplacian*.

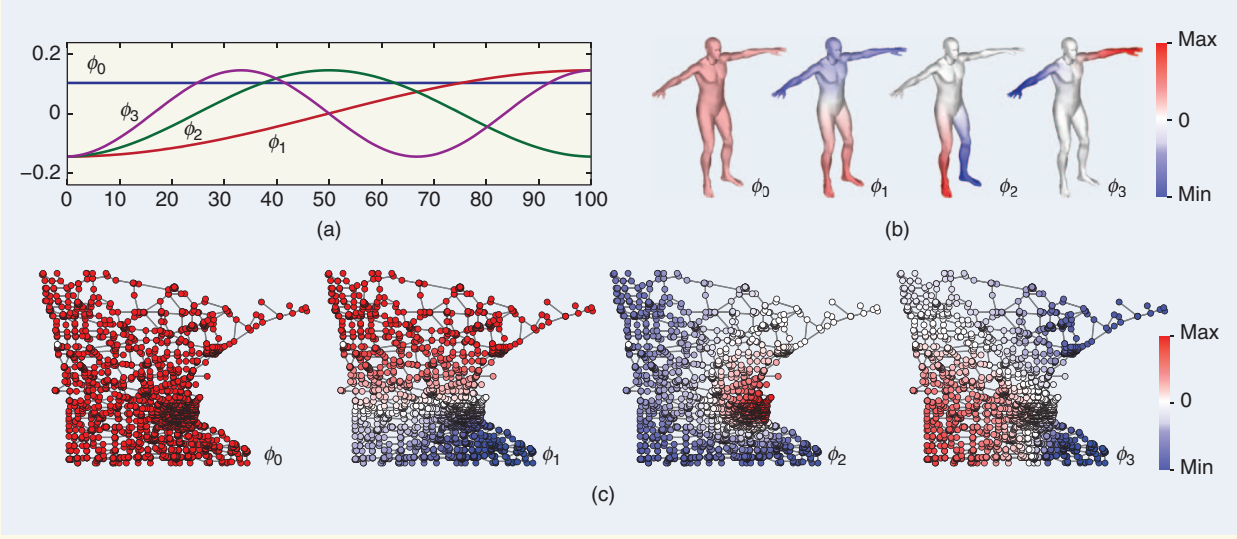



**FIGURE S2.** An example of the first four Laplacian eigenfunctions $\phi_0, \ldots, \phi_3$ on (a) a Euclidean domain (1-D line), and (b) and (c) non-Euclidean domains [(b) a human shape modeled as a 2-D manifold, and (c) a Minnesota road graph]. In the Euclidean case, the result is the standard Fourier basis comprising sinusoids of increasing frequency. In all cases, the eigenfunction $\phi_0$ corresponding to zero eigenvalue is constant (direct current component).1-D: one-dimensional.

operator in differential geometric jargon) $\Delta: L^2(\mathcal{X}) \to L^2(\mathcal{X})$ is an operator,

$$\Delta f = -\text{div}(\nabla f), \tag{14}$$

acting on scalar fields. Employing relation (13), it is easy to see that the Laplacian is self-adjoint (symmetric),

$$\langle \nabla f, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle \Delta f, f \rangle_{L^2(\mathcal{X})} = \langle f, \Delta f \rangle_{L^2(\mathcal{X})}. \tag{15}$$

The left-hand-side in (15) is known as the Dirichlet energy in physics and measures the smoothness of a scalar field on the manifold (see "Physical Interpretation of Laplacian Eigenfunctions"). The Laplacian can be interpreted as the difference between the average of a function on an infinitesimal

## Heat Diffusion on Non-Euclidean Domains

An important application of spectral analysis and, historically, the main motivation for its development by Joseph Fourier, is the solution of partial differential equations. Here, we are particularly interested in heat propagation on non-Euclidean domains. This process is governed by the heat diffusion equation, which in the simplest setting of homogeneous and isotropic diffusion has the form

$$\begin{cases} f_t(x,t) = -c\Delta f(x,t) \\ f(x,0) = f_0(x) \ \text{(Initial condition)} \end{cases} \tag{S5}$$

with additional boundary conditions if the domain has a boundary. $f(x,t)$ represents the temperature at point $x$ at time $t$. Equation (S5) encodes Newton's law of cooling, according to which the rate of temperature change of a body (left-hand side) is proportional to the difference between its own temperature and that of the surrounding

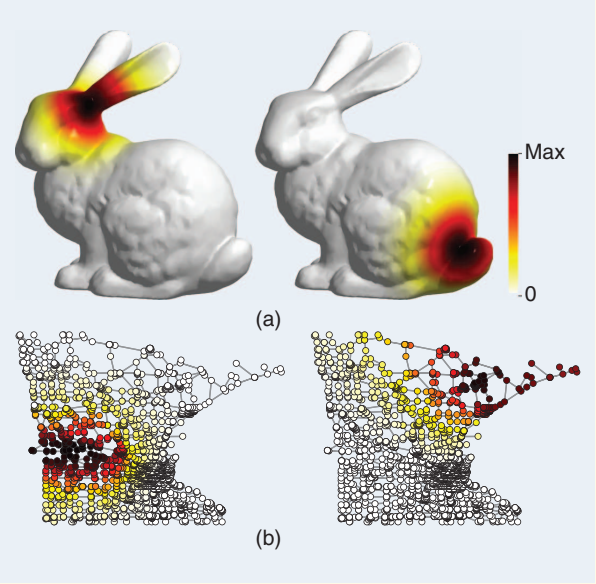



**FIGURE S3.** The examples of heat kernels on non-Euclidean domains [(a) manifold, and (b) graph]. Observe how moving the heat kernel to a different location changes its shape, which is an indication of the lack of shift invariance.

right-hand side. The proportion coefficient $c$ is referred to as the *thermal diffusivity constant*; here, we assume it to be equal to one for the sake of simplicity. The solution of (S5) is given by applying the heat operator $H^t = e^{-t\Delta}$ to the initial condition and can be expressed in the spectral domain as

$$\begin{aligned} f(x,t) = e^{-t\Delta} f_0(x) &= \sum_{i\geq 0} \langle f_0, \phi_i \rangle_{L^2(X)} e^{-t\lambda_i} \phi_i(x) \\ &= \int_X f_0(x') \underbrace{\sum_{i\geq 0} e^{-t\lambda_i} \phi_i(x) \phi_i(x')}_{h_t(x,x')} dx'. \end{aligned} \tag{S6}$$

$h_t(x,x')$ is known as the *heat kernel* (Figure S3) and represents the solution of the heat equation with an initial condition $f_0(x) = \delta_{x'}(x)$, or, in signal processing terms, an *impulse response*. In physical terms, $h_t(x,x')$ describes how much heat flows from a point $x$ to point $x'$ in time $t$. In the Euclidean case, the heat kernel is shift invariant, $h_t(x,x') = h_t(x-x')$, allowing to interpret the integral in (S6) as a convolution $f(x,t) = (f_0 \star h_t)(x)$. In the spectral domain, convolution with the heat kernel amounts to low-pass filtering with frequency response $e^{-t\lambda}$. Larger values of diffusion time $t$ result in lower effective cutoff frequency and thus smoother solutions in space (corresponding to the intuition that longer diffusion smoothes more the initial heat distribution).

The crosstalk between two heat kernels positioned at points $x$ and $x'$ allows to measure an intrinsic distance

$$d_t^2(x,x') = \int_X (h_t(x,y) - h_t(x',y))^2 dy \tag{S7}$$

$$= \sum_{i\geq 0} e^{-2t\lambda_i} (\phi_i(x) - \phi_i(x'))^2 \tag{S8}$$

referred to as the *diffusion distance* [30]. Note that when interpreting (S7) and (S8) as spatial- and frequency-domain norms $\|\cdot\|_{L^2(X)}$ and $\|\cdot\|_{\ell^2}$, respectively, their equivalence is the consequence of the Parseval identity. Unlike geodesic distance that measures the length of the shortest path on the manifold or graph, the diffusion distance has an effect of averaging over different paths. It is thus more robust to perturbations of the domain, e.g., introduction or removal of edges in a graph or cuts on a manifold.

sphere around a point and the value of the function at the point itself. It is one of the most important operators in mathematical physics, used to describe phenomena as diverse as heat diffusion (see "Heat Diffusion on Non-Euclidean Domains"), quantum mechanics, and wave propagation. As we will see in the following, the Laplacian plays a central role in signal processing and learning on non-Euclidean domains, as its eigenfunctions generalize the classical Fourier bases, allowing to perform spectral analysis on manifolds and graphs.

It is important to note that all the previous definitions are coordinate free. By defining a basis in the tangent space, it is possible to express tangent vectors as $d$-dimensional vectors and the Riemannian metric as a $d \times d$ symmetric positive-definite matrix.

### *Graphs and discrete differential operators*

Another type of constructions we are interested in are graphs, which are popular models of networks, interactions, and

similarities between different objects. For simplicity, we will consider weighted undirected graphs, formally defined as a pair $(\mathcal{V},\mathcal{E})$, where $\mathcal{V}=\{1,\ldots,n\}$ is the set of $n$ vertices, and $\mathcal{E}\subseteq\mathcal{V}\times\mathcal{V}$ is the set of edges, where the graph being undirected implies that $(i,j)\in\mathcal{E}$ if $(j,i)\in\mathcal{E}$. Furthermore, we associate a weight $a_i>0$ with each vertex $i\in\mathcal{V}$, and a weight $w_{ij}\geq 0$ with each edge $(i,j)\in\mathcal{E}$.

Real functions $f:\mathcal{V}\to\mathbb{R}$ and $F:\mathcal{E}\to\mathbb{R}$ on the vertices and edges of the graph, respectively, are roughly the discrete analogy of continuous scalar and tangent vector fields in differential geometry (it is tacitly assumed here that $F$ is alternating, i.e., $F_{ij}=-F_{ji}$). We define Hilbert spaces $L^2(\mathcal{V})$ and $L^2(\mathcal{E})$ of such functions by specifying the respective inner products,

$$\langle f,g\rangle_{L^2(\mathcal{V})}=\sum_{i\in\mathcal{V}}a_i f_i g_i, \tag{16}$$

$$\langle F,G\rangle_{L^2(\mathcal{E})}=\sum_{i\in\mathcal{E}}w_{ij}F_{ij}G_{ij}. \tag{17}$$

Let $f\in L^2(\mathcal{V})$ and $F\in L^2(\mathcal{E})$ be functions on the vertices and edges of the graphs, respectively. We can define differential operators acting on such functions analogously to differential operators on manifolds [72]. The graph gradient is an operator $\nabla:L^2(\mathcal{V})\to L^2(\mathcal{E})$ mapping functions defined on vertices to functions defined on edges,

$$(\nabla f)_{ij}=f_i-f_j, \tag{18}$$

automatically satisfying $(\nabla f)_{ij}=-(\nabla f)_{ji}$. The graph divergence is an operator $\mathrm{div}:L^2(\mathcal{E})\to L^2(\mathcal{V})$ doing the converse,

$$(\mathrm{div}F)_i=\frac{1}{a_i}\sum_{j:(i,j)\in\mathcal{E}}w_{ij}F_{ij}. \tag{19}$$

It is easy to verify that the two operators are adjoint with respect to the inner products (16) and (17),

$$\langle F,\nabla f\rangle_{L^2(\mathcal{E})}=\langle\nabla^* F,f\rangle_{L^2(\mathcal{V})}=\langle-\mathrm{div}F,f\rangle_{L^2(\mathcal{V})}. \tag{20}$$

The graph Laplacian is an operator $\Delta:L^2(\mathcal{V})\to L^2(\mathcal{V})$ defined as $\Delta=-\mathrm{div}\nabla$. Combining definitions (18) and (19), it can be expressed in the familiar form

$$(\Delta f)_i=\frac{1}{a_i}\sum_{(i,j)\in\mathcal{E}}w_{ij}(f_i-f_j). \tag{21}$$

Note that (21) captures the intuitive geometric interpretation of the Laplacian as the difference between the local average of a function around a point and the value of the function at the point itself.

Denoting by $\mathbf{W}=(w_{ij})$ the $n\times n$ matrix of edge weights [it is assumed that $w_{ij}=0$ if $(i,j)\notin\mathcal{E}$], by $\mathbf{A}=\mathrm{diag}(a_1,\ldots,a_n)$ the diagonal matrix of vertex weights, and by $\mathbf{D}=\mathrm{diag}\left(\sum_{j:j\neq i}w_{ij}\right)$ the degree matrix, the graph Laplacian application to a function $f\in L^2(\mathcal{V})$ represented as a column vector $\mathbf{f}=(f_1,\ldots,f_n)^\top$ can be written in matrix-vector form as

$$\Delta\mathbf{f}=\mathbf{A}^{-1}(\mathbf{D}-\mathbf{W})\mathbf{f}. \tag{22}$$

The choice of $\mathbf{A}=\mathbf{I}$ in (22) is referred to as the *unnormalized graph Laplacian*; another popular choice is $\mathbf{A}=\mathbf{D}$ producing the random walk Laplacian [73].

### Discrete manifolds

As previously mentioned, there are many practical situations in which one is given a sampling of points arising from a manifold but not the manifold itself. In computer graphics applications, reconstructing a correct discretization of a manifold from a point cloud is a difficult problem of its own, referred to as *meshing* (see "Laplacian on Discrete Manifolds"). In manifold-learning problems, the manifold is typically approximated as a graph capturing the local affinity structure. We stress that the term *manifold* as used in the context of generic data science is not geometrically rigorous and can have less structure than a classical smooth manifold we have defined beforehand. For example, a set of points that looks locally Euclidean in practice may have self-intersections, infinite curvature, different dimensions depending on the scale and location at which one looks, extreme variations in density, and noise with confounding structure.

### Fourier analysis on non-Euclidean domains

The Laplacian operator is a self-adjoint positive-semidefinite operator, admitting on a compact domain an eigendecomposition with a discrete set of orthonormal eigenfunctions $\phi_0,\phi_1,\ldots$ (satisfying $\langle\phi_i,\phi_j\rangle_{L^2(\mathcal{X})}=\delta_{ij}$) and nonnegative real eigenvalues $0=\lambda_0\leq\lambda_1\leq\ldots$ (referred to as the *spectrum* of the Laplacian),

$$\Delta\phi_i=\lambda_i\phi_i, i=0,1,\ldots. \tag{23}$$

[Note that in the Euclidean case, the Fourier transform of a function defined on a finite interval (which is a compact set) or its periodic extension is discrete. In practical settings, all domains we are dealing with are compact.]

The eigenfunctions are the smoothest functions in the sense of the Dirichlet energy (see "Physical Interpretation of Laplacian Eigenfunctions") and can be interpreted as a generalization of the standard Fourier basis [given, in fact, by the eigenfunctions of the one-dimensional (1-D) Euclidean Laplacian, $-\left(d^2/x^2\right)e^{i\omega x}=\omega^2 e^{i\omega x}$] to a non-Euclidean domain. It is important to emphasize that the Laplacian eigenbasis is intrinsic due to the intrinsic construction of the Laplacian itself.

A square-integrable function $f$ on $\mathcal{X}$ can be decomposed into Fourier series as

$$f(x)=\sum_{i\geq 0}\underbrace{\langle f,\phi_i\rangle_{L^2(\mathcal{X})}}_{\hat{f}_i}\phi_i(x), \tag{24}$$

where the projection on the basis functions producing a discrete set of Fourier coefficients $(\hat{f}_0,\hat{f}_1,\ldots)$ generalizes the analysis (forward transform) stage in classical signal processing,

## Laplacian on Discrete Manifolds

In computer graphics and vision applications, 2-D manifolds are commonly used to model 3-D shapes. There are several common ways of discretizing such manifolds. First, the manifold is assumed to be sampled at $n$ points. Their embedding coordinates $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are referred to as a *point cloud*. Second, a graph is constructed upon these points, acting as its vertices. The edges of the graph represent the local connectivity of the manifold, telling whether two points belong to a neighborhood or not. The graph can be endowed, e.g., with Gaussian-edge weights

$$w_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2}. \tag{S9}$$

This simplest discretization, however, does not correctly capture the geometry of the underlying continuous manifold (e.g., the graph Laplacian would typically not converge to the continuous Laplacian operator of the manifold with the increase of the sampling density [68]). A geometrically consistent discretization is possible with an additional structure of faces $\mathcal{F} \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$, where $(i,j,k) \in \mathcal{F}$ implies $(i,j),(i,k),(k,j) \in \mathcal{E}$. The collection of faces represents the underlying continuous manifold as a polyhedral surface consisting of small triangles glued together. The triplet $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ is referred to as *triangular mesh*. To be a correct discretization of a manifold (a manifold mesh), every edge must be shared by exactly two triangular faces; if the manifold has a boundary, any boundary edge must belong to exactly one triangle.

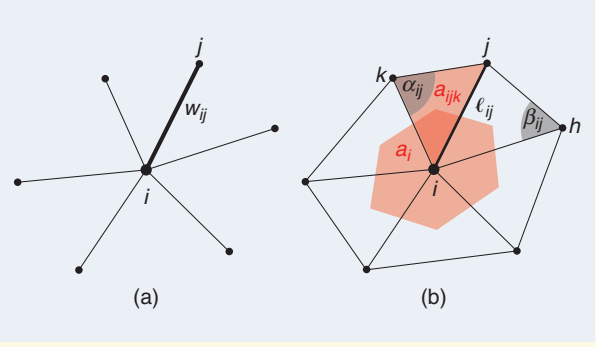



**FIGURE S4.** The two commonly used discretizations of a 2-D manifold: (a) an undirected graph and (b) a triangular mesh.

On a triangular mesh, the simplest discretization of the Riemannian metric is given by assigning each edge a length $\ell_{ij} > 0$, which must additionally satisfy the triangle inequality in every triangular face. The mesh Laplacian is given by (21) with

$$w_{ij} = \frac{-\ell_{ij}^2 + \ell_{jk}^2 + \ell_{ik}^2}{8a_{ijk}} + \frac{-\ell_{ij}^2 + \ell_{jh}^2 + \ell_{ih}^2}{8a_{ijh}}; \tag{S10}$$

$$a_i = \frac{1}{3} \sum_{jk:(i,j,k) \in \mathcal{F}} a_{ijk}, \tag{S11}$$

where $a_{ijk} = \sqrt{s_{ijk}(s_{ijk} - \ell_{ij})(s_{ijk} - \ell_{jk})(s_{ijk} - \ell_{ik})}$ is the area of triangle $ijk$ given by the Heron formula, and $s_{ijk} = (1/2)(\ell_{ij} + \ell_{jk} + \ell_{ki})$ is the semiperimeter of triangle $ijk$. The vertex weight $a_i$ is interpreted as the local area element (shown in red in Figure S4). Note that the weights (S10) and (S11) are expressed solely in terms of the discrete metric $\ell$ and are thus intrinsic. When the mesh is infinitely refined under some technical conditions, such a construction can be shown to converge to the continuous Laplacian of the underlying manifold [69].

An embedding of the mesh (amounting to specifying the vertex coordinates $\mathbf{x}_1, \ldots, \mathbf{x}_n$) induces a discrete metric $\ell_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$, whereby (S10) become the cotangent weights

$$w_{ij} = \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) \tag{S12}$$

ubiquitously used in computer graphics [70].

and summing up the basis functions with these coefficients is the synthesis (inverse transform) stage.

A centerpiece of classical Euclidean signal processing is the property of the Fourier transform diagonalizing the convolution operator, colloquially referred to as the *convolution theorem*. This property allows to express the convolution $f \star g$ of two functions in the spectral domain as the elementwise product of their Fourier transforms,

$$\widehat{(f \star g)}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx \int_{-\infty}^{\infty} g(x) e^{-i\omega x} dx. \tag{25}$$

Unfortunately, in the non-Euclidean case, we cannot even define the operation $x - x'$ on the manifold or graph, so the notion of convolution (7) does not directly extend to this case. One possibility to generalize convolution to non-Euclidean domains is by using the convolution theorem as a definition,

$$(f \star g)(x) = \sum_{i \geq 0} \langle f, \phi_i \rangle_{L^2(\mathcal{X})} \langle g, \phi_i \rangle_{L^2(\mathcal{X})} \phi_i(x). \tag{26}$$

One of the key differences of such a construction from the classical convolution is the lack of shift invariance. In terms of signal processing, it can be interpreted as a position-dependent filter. While parameterized by a fixed number of coefficients in the frequency domain, the spatial representation of the filter can vary dramatically at different points (see Figure S3).

In situations where the graph is constructed from the data, a straightforward choice of the edge weights (S9) of the graph is the covariance of the data Let **F** denote the input data distribution and

$$\mathbf{\Sigma} = \mathbb{E}(\mathbf{F} - \mathbb{E}\mathbf{F})(\mathbf{F} - \mathbb{E}\mathbf{F})^\top \tag{S13}$$

be the data covariance matrix. If each point has the same variance $\sigma_{ii} = \sigma^2$, then diagonal operators on the Laplacian simply scale the principal components of **F**.

In natural images, because their distribution is approximately stationary, the covariance matrix has a circulant structure $\sigma_{ij} \approx \sigma_{i-j}$ and is thus diagonalized by the standard discrete cosine transform (DCT) basis. It follows that the principal components of **F** roughly correspond to the DCT basis vectors ordered by frequency. Moreover, natural images exhibit a power spectrum $\mathbb{E}|\hat{f}(\omega)|^2 \sim |\omega|^{-2}$, because nearby pixels are more correlated than faraway pixels [14]. It results that principal components of the covariance are essentially ordered from low to high frequencies, which is consistent with the standard group structure of the Fourier basis. When applied to natural images represented as graphs with weights defined by the covariance, the SCNN construction recovers the standard CNN, without any prior knowledge [76] (Figure S5). Indeed, the linear operators $\mathbf{\Phi\Gamma}_{l,l'}\mathbf{\Phi}^\top$ in (27) are by the previous argument diagonal in the Fourier basis, hence translation invariant, hence classical convolutions. Furthermore, the "Spectrum-Free Methods" section explains how spatial subsampling can also be obtained via dropping the last part of the spectrum of the Laplacian, leading to pooling, and ultimately to standard CNNs.

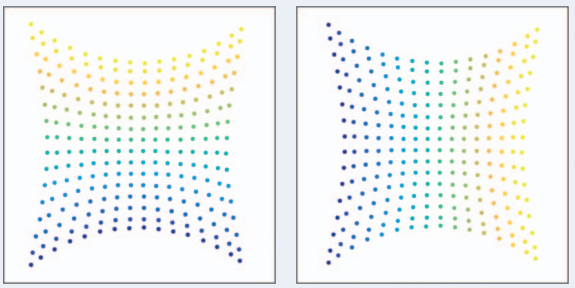



**FIGURE S5.** The 2-D embedding of pixels in 16 × 16 image patches using a Euclidean radial basis function (RBF) kernel. The RBF kernel is constructed as in (S9), by using the covariance $\sigma_{ij}$ as Euclidean distance between two features. The pixels are embedded in a 2-D space using the first two eigenvectors of the resulting graph Laplacian. The colors in (a) and (b) represent the horizontal and vertical coordinates of the pixels, respectively. The spatial arrangement of pixels is roughly recovered from correlation measurements.

The previous discussion also applies to graphs instead of manifolds, where one only has to replace the inner product in (24) and (26) with the discrete one (16). All of the sums over $i$ would become finite, as the graph Laplacian matrix $\Delta$ has $n$ eigenvectors. In matrix-vector notation, the generalized convolution $f \star g$ can be expressed as $\mathbf{Gf} = \mathbf{\Phi}\,\mathrm{diag}(\hat{\mathbf{g}})\mathbf{\Phi}^\top\mathbf{f}$, where $\hat{\mathbf{g}} = (\hat{g}_1, \ldots, \hat{g}_n)$ is the spectral representation of the filter, and $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ denotes the Laplacian eigenvectors (S8). The lack of shift invariance results in the absence of circulant (Toeplitz) structure in the matrix **G**, which characterizes the Euclidean setting. Furthermore, it is easy to see that the convolution operation commutes with the Laplacian, $\mathbf{G}\Delta\mathbf{f} = \Delta\mathbf{Gf}$.

## *Uniqueness and stability*

Finally, it is important to note that the Laplacian eigenfunctions are not uniquely defined. To start with, they are defined up to sign, i.e., $\Delta(\pm\phi) = \lambda(\pm\phi)$. Thus, even isometric domains might have different Laplacian eigenfunctions. Furthermore, if a Laplacian eigenvalue has multiplicity, then the associated eigenfunctions can be defined as orthonormal basis spanning the corresponding eigensubspace (or said differently, the eigenfunctions are defined up to an orthogonal transformation in the

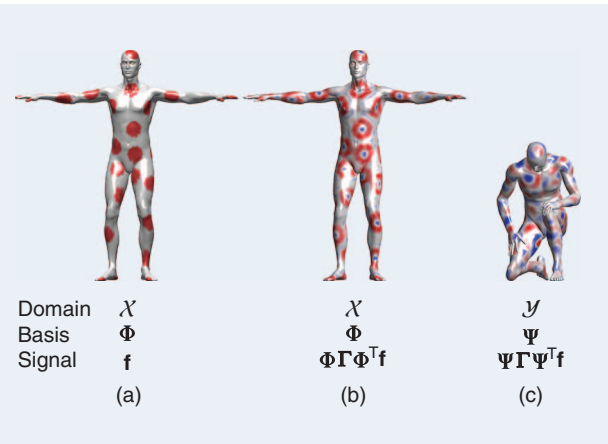



**FIGURE 2.** A toy example illustrating the difficulty of generalizing spectral filtering across non-Euclidean domains. (a) A function defined on a manifold (function values are represented by color). (b) The result of the application of an edge-detection filter in the frequency domain. (c) The same filter applied on the same function but on a different (nearly isometric) domain produces a completely different result. The reason for this behavior is that the Fourier basis is domain dependent and the filter coefficients learned on one domain cannot be applied to another one in a straightforward manner.

## Citation Network Analysis Application

The CORA citation network [90] is a graph containing 2,708 vertices representing articles and 5,429 edges representing citations (Figure S6). Each article is described by a 1,433-dimensional bag-of-words feature vector and belongs to seven classes. For simplicity, the network is treated as an undirected graph. Applying the SCNN with two spectral convolutional layers parameterized according to (37), the authors of [77] obtained classification accuracy of 81.6% (compared to the previous best result of 75.7%). In [54], this result was slightly improved further, reaching 81.7% accuracy with the use of MoNet architecture.

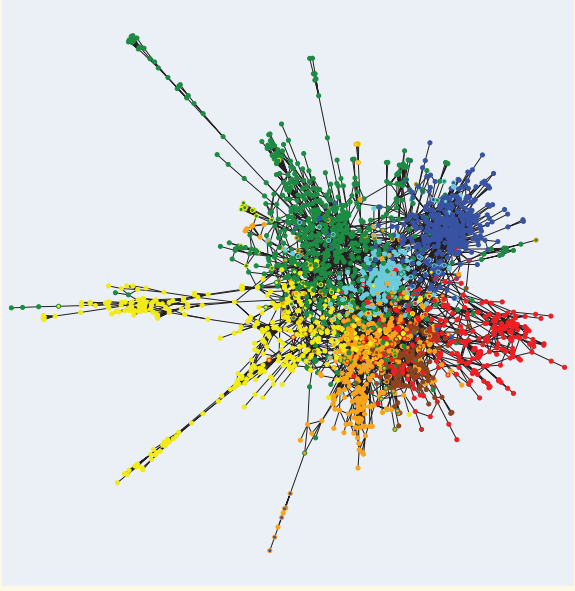


**FIGURE S6.** The classifying of a research article in the CORA data set with MoNet. Shown is the citation graph, where each node is an article and an edge represents a citation. Vertex fill and outline colors represent the predicted and ground-truth labels, respectively; ideally, the two colors should coincide. (Figure reproduced from [54].)

eigensubspace). A small perturbation of the domain can lead to very large changes in the Laplacian eigenvectors, especially those associated with high frequencies. At the same time, the definition of heat kernels (S6) and diffusion distances (S8) does not suffer from these ambiguities, e.g., the sign ambiguity disappears as the eigenfunctions are squared. Heat kernels also appear to be robust to domain perturbations.

## Spectral methods

We have now finally gotten to our main goal, namely, constructing a generalization of the CNN architecture on non-Euclidean domains. We will start with the assumption that the domain on which we are working is fixed, and for the rest of this section, we will use the problem of classification of a signal on a fixed graph as the prototypical application. We have seen that convolutions are linear operators that commute with the Laplacian operator. Therefore, given a weighted graph, a first route to generalize a convolutional architecture is by first restricting our interest to linear operators that commute with the graph Laplacian. This property, in turn, implies operating on the spectrum of the graph weights, given by the eigenvectors of the graph Laplacian.

### *Spectral CNN*

Similarly to the convolutional layer (6) of a classical Euclidean CNN, Bruna et al. [52] define a spectral convolutional layer as

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{q} \boldsymbol{\Phi}_k \boldsymbol{\Gamma}_{l,l'} \boldsymbol{\Phi}_k^\top \mathbf{f}_{l'}\right), \tag{27}$$

where the $n \times p$ and $n \times q$ matrices $\mathbf{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_p)$ and $\mathbf{G} = (\mathbf{g}_1, \ldots, \mathbf{g}_q)$ represent the $p$- and $q$-dimensional input and output signals on the vertices of the graph, respectively (we use $n = |\mathcal{V}|$ to denote the number of vertices in the graph), $\boldsymbol{\Gamma}_{l,l'}$ is a $k \times k$ diagonal matrix of spectral multipliers representing a filter in the frequency domain, and $\xi$ is a nonlinearity applied on the vertex-wise function values. Using only the first $k$ eigenvectors in (27) sets a cutoff frequency that depends on the intrinsic regularity of the graph and also the sample size. Typically, $k \ll n$, because only the first Laplacian eigenvectors describing the smooth structure of the graph are useful in practice.

If the graph has an underlying group invariance, such a construction can discover it. In particular, standard CNNs can be redefined from the spectral domain (see "Rediscovering Standard CNNs Using Correlation Kernels"). However, in many cases the graph does not have a group structure, or the group structure does not commute with the Laplacian, and so we cannot think of each filter as passing a template across $\mathcal{V}$ and recording the correlation of the template with that location.

We should stress that a fundamental limitation of the spectral construction is its restriction to a single domain. The reason is that spectral filter coefficients (27) are basis dependent. It implies that if we learn a filter with respect to basis $\boldsymbol{\Phi}_k$ on one domain, and then try to apply it on another domain with another basis $\boldsymbol{\Psi}_k$, the result could be very different (see Figure 2). It is possible to construct compatible orthogonal bases across different domains resorting to a joint diagonalization procedure [74], [75]. However, such a construction requires the knowledge of some correspondence between the domains. In applications like social network analysis, e.g., where dealing with two time instances of a social graph in which new vertices and edges have been added, such a correspondence can be easily computed and is therefore a reasonable assumption. Conversely, in computer graphics applications, finding correspondence between shapes is in itself a very hard problem, so assuming known correspondence between the domains is a rather unreasonable assumption.

Assuming that $k = O(n)$ eigenvectors of the Laplacian are kept, a convolutional layer (27) requires $pqk = O(n)$ parameters to train. We will see next how the global and local regularity of the graph can be combined to produce layers with constant number of parameters (i.e., such that the number of learnable parameters per layer does not depend upon the size of the input), which is the case in classical Euclidean CNNs.

The non-Euclidean analogy of pooling is graph coarsening, in which only a fraction $\alpha < 1$ of the graph vertices is retained. The eigenvectors of graph Laplacians at two different resolutions are related by the following multigrid property: let $\mathbf{\Phi}, \tilde{\mathbf{\Phi}}$ denote the $n \times n$ and $\alpha n \times \alpha n$ matrices of Laplacian eigenvectors of the original and the coarsened graph, respectively. Then,

$$\tilde{\mathbf{\Phi}} \approx \mathbf{P}\mathbf{\Phi}\begin{pmatrix}\mathbf{I}_{\alpha n}\\ 0\end{pmatrix}, \tag{28}$$

where $\mathbf{P}$ is an $\alpha n \times n$ binary matrix whose $i$th row encodes the position of the $i$th vertex of the coarse graph on the original graph. It follows that strided convolutions can be generalized using the spectral construction by keeping only the low-frequency components of the spectrum. This property also allows us to interpret (via interpolation) the local filters at deeper layers in the spatial construction to be low frequency. However, because in (27) the nonlinearity is applied in the spatial domain, in practice one has to recompute the graph Laplacian eigenvectors at each resolution and apply them directly after each pooling step.

The spectral construction (27) assigns a DoF for each eigenvector of the graph Laplacian. In most graphs, individual high-frequency eigenvectors become highly unstable. However, similarly as the wavelet construction in Euclidean domains, by appropriately grouping high-frequency eigenvectors in each octave, one can recover meaningful and stable information. As shown next, this principle also entails better learning complexity.

### *Spectral CNN with smooth spectral multipliers*

To reduce the risk of overfitting, it is important to adapt the learning complexity to reduce the number of free parameters of the model [44], [52]. On Euclidean domains, this is achieved by learning convolutional kernels with small spatial support, which enables the model to learn a number of parameters independent of the input size. To achieve a similar learning complexity in the spectral domain, it is thus necessary to restrict the class of spectral multipliers to those corresponding to localized filters.

For that purpose, we have to express spatial localization of filters in the frequency domain. In the Euclidean case, smoothness in the frequency domain corresponds to spatial decay, because

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega, \tag{29}$$

by virtue of the Parseval identity. This suggests that, to learn a layer in which features will be not only shared across locations but also well localized in the spatial domain, one can learn spectral multipliers that are smooth. Smoothness can be prescribed by learning only a subsampled set of spectral multipliers and using an interpolation kernel to obtain the rest, such as cubic splines.

However, the notion of smoothness also requires some geometry in the spectral domain. In the Euclidean setting, such a geometry naturally arises from the notion of frequency, e.g., in the plane, the similarity between two Fourier atoms $e^{i\boldsymbol{\omega}^\top \mathbf{x}}$ and $e^{i\boldsymbol{\omega}'^\top \mathbf{x}}$ can be quantified by the distance $\|\boldsymbol{\omega} - \boldsymbol{\omega}'\|$, where $\mathbf{x}$ denotes the 2-D planar coordinates, and $\boldsymbol{\omega}$ is the 2-D frequency vector. On graphs, such a relation can be defined by means of a dual graph with weights $\tilde{w}_{ij}$ encoding the similarity between two eigenvectors $\phi_i$ and $\phi_j$.

A particularly simple choice consists in choosing a 1-D arrangement, obtained by ordering the eigenvectors according to their eigenvalues. [In the mentioned 2-D example, this would correspond to ordering the Fourier basis function according to the sum of the corresponding frequencies $\omega_1 + \omega_2$. Although numerical results on simple low-dimensional graphs show that the 1-D arrangement given by the spectrum of the Laplacian is efficient at creating spatially localized filters [52], an open fundamental question is how to define a dual graph on the eigenvectors of the Laplacian in which smoothness (obtained by applying the diffusion operator) corresponds to localization in the original graph.] In this setting, the spectral multipliers are parameterized as

$$\mathrm{diag}(\mathbf{\Gamma}_{l,l'}) = \mathbf{B}\boldsymbol{\alpha}_{l,l'}, \tag{30}$$

where $\mathbf{B} = (b_{ij}) = (\beta_j(\lambda_i))$ is a $k \times q$ fixed interpolation kernel [e.g., $\beta_j(\lambda)$ can be cubic splines], and $\boldsymbol{\alpha}$ is a vector of $q$ interpolation coefficients. To obtain filters with constant spatial support (i.e., independent of the input size $n$), one should choose a sampling step $\gamma \sim n$ in the spectral domain, which results in a constant number $n\gamma^{-1} = O(1)$ of coefficients $\boldsymbol{\alpha}_{l,l'}$ per filter. Therefore, by combining spectral layers with graph coarsening, this model has $O(\log n)$ total trainable parameters for inputs of size $n$, thus recovering the same learning complexity as CNNs on Euclidean grids.

Even with such a parameterization of the filters, the spectral CNN (27) entails a high computational complexity of performing forward and backward passes, because they require an expensive step of matrix multiplication by $\mathbf{\Phi}_k$ and $\mathbf{\Phi}_k^\top$. While on Euclidean domains such a multiplication can be efficiently carried in $O(n \log n)$ operations using fast-Fourier-transform-type algorithms, for general graphs such algorithms do not exist and the complexity is $O(n^2)$. We will see next how to alleviate this cost by avoiding explicit computation of the Laplacian eigenvectors.

## Spectrum-free methods

A polynomial of the Laplacian acts as a polynomial on its eigenvalues. Thus, instead of explicitly operating in the frequency

domain with spectral multipliers as in (30), it is possible to represent the filters via a polynomial expansion:

$$g_\alpha(\Delta) = \mathbf{\Phi} g_\alpha(\mathbf{\Lambda})\mathbf{\Phi}^\top, \tag{31}$$

where

$$g_\alpha(\lambda) = \sum_{j=0}^{r-1} \alpha_j \lambda^j, \tag{32}$$

$\boldsymbol{\alpha}$ is the $r$-dimensional vector of polynomial coefficients, and $g_\alpha(\mathbf{\Lambda}) = \mathrm{diag}(g_\alpha(\lambda_1), \ldots, g_\alpha(\lambda_n))$, resulting in filter matrices $\mathbf{\Gamma}_{l,l'} = g_{\alpha_{l,l'}}(\mathbf{\Lambda})$ whose entries have an explicit form in terms of the eigenvalues.

An important property of this representation is that it automatically yields localized filters, for the following reason. Because the Laplacian is a local operator (working on one-hop neighborhoods), the action of its $j$th power is constrained to $j$ hops. Because the filter is a linear combination of powers of the Laplacian, overall (32) behaves like a diffusion operator limited to $r$ hops around each vertex.

### GCNN, also known as ChebNet

Defferrard et al. used the Chebyshev polynomials generated by the recurrence relation [45]

$$\begin{aligned} T_j(\lambda) &= 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda), \\ T_0(\lambda) &= 1, \\ T_1(\lambda) &= \lambda. \end{aligned} \tag{33}$$

A filter (32) can thus be parameterized uniquely via an expansion of order $r-1$ such that

$$\begin{aligned} g_\alpha(\tilde{\Delta}) &= \sum_{j=0}^{r-1} \alpha_j \mathbf{\Phi} T_j(\tilde{\mathbf{\Lambda}})\mathbf{\Phi}^\top \\ &= \sum_{j=0}^{r-1} \alpha_j T_j(\tilde{\Delta}), \end{aligned} \tag{34}$$

where $\tilde{\Delta} = 2\lambda_n^{-1}\Delta - \mathbf{I}$ and $\tilde{\mathbf{\Lambda}} = 2\lambda_n^{-1}\mathbf{\Lambda} - \mathbf{I}$ denotes a rescaling of the Laplacian mapping its eigenvalues from the interval $[0, \lambda_n]$ to $[-1, 1]$ (necessary because the Chebyshev polynomials form an orthonormal basis in $[-1, 1]$).

Denoting $\bar{\mathbf{f}}^{(j)} = T_j(\tilde{\Delta})\mathbf{f}$, we can use the recurrence relation (33) to compute $\bar{\mathbf{f}}^{(j)} = 2\tilde{\Delta}\bar{\mathbf{f}}^{(j-1)} - \bar{\mathbf{f}}^{(j-2)}$, with $\bar{\mathbf{f}}^{(0)} = \mathbf{f}$ and $\bar{\mathbf{f}}^{(1)} = \tilde{\Delta}\mathbf{f}$. The computational complexity of this procedure is therefore $O(rn)$ operations and does not require an explicit computation of the Laplacian eigenvectors.

### Graph convolutional network

Kipf and Welling [77] simplified this construction by further assuming $r \approx 2$ and $\lambda_n \approx 2$, resulting in filters of the form

$$\begin{aligned} g_\alpha(\mathbf{f}) &= \alpha_0 \mathbf{f} + \alpha_1(\Delta - \mathbf{I})\mathbf{f} \\ &= \alpha_0 \mathbf{f} - \alpha_1 \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f}. \end{aligned} \tag{35}$$

Further constraining $\alpha = \alpha_0 = -\alpha_1$, one obtains filters represented by a single parameter,

$$g_\alpha(\mathbf{f}) = \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}. \tag{36}$$

Because the eigenvalues of $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ are now in the range $[0, 2]$, repeated application of such a filter can result in numerical instability. This can be remedied by a renormalization

$$g_\alpha(\mathbf{f}) = \alpha\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{f}, \tag{37}$$

where $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathrm{diag}\left(\sum_{j \neq i} \tilde{w}_{ij}\right)$.

Note that though we arrived at the constructions of ChebNet and graph convolutional network (GCN) starting in the spectral domain, they boil down to applying simple filters acting on the $r$- or one-hop neighborhood of the graph in the spatial domain. We consider these constructions to be examples of the more general graph neural network (GNN) framework.

### GNN

GNNs [78] generalize the notion of applying the filtering operations directly on the graph via the graph weights. Similarly as Euclidean CNNs learn generic filters as linear combinations of localized, oriented bandpass and low-pass filters, a GNN learns at each layer a generic linear combination of graph low-pass and high-pass operators. These are given, respectively, by $f \mapsto \mathbf{W}f$ and $f \mapsto \Delta f$ and are thus generated by the degree matrix $\mathbf{D}$ and the diffusion matrix $\mathbf{W}$. Given a $p$-dimensional input signal on the vertices of the graph, represented by the $n \times p$ matrix $\mathbf{F}$, the GNN considers a generic nonlinear function $\eta_{\boldsymbol{\theta}}: \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^q$, parameterized by trainable parameters $\boldsymbol{\theta}$ that is applied to all nodes of the graph,

$$\mathbf{g}_i = \eta_{\boldsymbol{\theta}}((\mathbf{W}\mathbf{f})_i, (\mathbf{D}\mathbf{f})_i). \tag{38}$$

In particular, choosing $\eta(\mathbf{a}, \mathbf{b}) = \mathbf{b} - \mathbf{a}$, one recovers the Laplacian operator $\Delta\mathbf{f}$, but more general, nonlinear choices for $\eta$ yield trainable, task-specific diffusion operators. Similarly as with a CNN architecture, one can stack the resulting GNN layers $\mathbf{g} = C_{\boldsymbol{\theta}}(\mathbf{f})$ and interleave them with graph pooling operators. Chebyshev polynomials $T_r(\Delta)$ can be obtained with $r$ layers of (38), making it possible, in principle, to consider ChebNet and GCN as particular instances of the GNN framework.

Historically, a version of GNN was the first formulation of deep learning on graphs, proposed in [49] and [78]. These works optimized over the parameterized steady state of some diffusion process (or random walk) on the graph. This can be interpreted as in (38) but using a large number of layers where each $C_{\boldsymbol{\theta}}$ is identical, as the forward propagation through the $C_{\boldsymbol{\theta}}$ approximate the steady state. Recent works [50], [51], [55], [79], [80] relax the requirements of approaching the steady state or using repeated applications of the same $C_{\boldsymbol{\theta}}$.

Because the communication at each layer is local to a vertex neighborhood, one may worry that it would take many layers to get information from one part of the graph to another, requiring multiple hops (this was one of the reasons for the use of the steady state in [78]). However, for many applications, it is not necessary for information to completely traverse

## Three-Dimensional Shape Correspondence Application

Finding intrinsic correspondence between deformable shapes is a classical tough problem that underlies a broad range of vision and graphics applications, including texture mapping, animation, editing, and scene understanding [107]. From the machine-learning standpoint, correspondence can be thought of as a classification problem, where each point on the query shape is assigned to one of the points on a reference shape (serving as a label space) [108]. It is possible to learn the correspondence with a deep intrinsic network applied to some input feature vector $\mathbf{f}(x)$ at each point $x$ of the query shape $\mathcal{X}$, producing an output $U_\Theta(\mathbf{f}(x))(y)$, which is interpreted as the conditional probability $p(y \mid x)$ of $x$ being mapped to $y$ [Figure S7(a)]. Using a training set of points with their ground-truth correspondence $\{x_i, y_i\}_{i \in \mathcal{I}}$, supervised learning is performed minimizing the multinomial regression loss

$$\min_\Theta - \sum_{i \in \mathcal{I}} \log U_\Theta(\mathbf{f}(x_i))(y_i) \tag{S14}$$

with respect to the network parameters $\Theta$. The loss penalizes for the deviation of the predicted correspondence from the ground truth. We note that, while producing impressive results [Figure S7(b)], such an approach essentially learns pointwise correspondence, which then has to be postprocessed to satisfy certain properties, such as smoothness or bijectivity. Correspondence is an example of structured output, where the output of the network at one point depends on the output in other points (in the simplest setting, correspondence should be smooth, i.e., the output at nearby points should be similar) Litany et al. [109] proposed intrinsic structured prediction of shape correspondence by integrating a layer computing functional correspondence [106] into the deep neural network.


**FIGURE S7.** (a) The learning shape correspondence: an intrinsic deep network $U_\Theta$ is applied pointwise to some input features defined at each point. The output of the network at each point $x$ of the query shape $\mathcal{X}$ is a probability distribution of the reference shape $\mathcal{Y}$ that can be thought of as a soft correspondence. (b) The intrinsic correspondence established between human shapes using intrinsic deep architecture (MoNet [54] with three convolutional layers). Signature of histogram orientations (SHOT) descriptors capturing the local normal vector orientations [110] were used in this example as input features. The correspondence is visualized by transferring texture from the leftmost reference shape. For additional examples, see [54].

the graph. Furthermore, note that the graphs at each layer of the network need not be the same. Thus, we can replace the original neighborhood structure with one's favorite multiscale coarsening of the input graph and operate on that to obtain the same flow of information as with the convolutional nets above (or rather more like a locally connected network [81]). This also allows producing a single output for the whole graph (for translation-invariant tasks), rather than a per-vertex output, by connecting each vertex to a special output node. Alternatively, one can allow $\eta$ to use not only $\mathbf{Wf}$ and $\Delta\mathbf{f}$ at each node but also $\mathbf{W}^s\mathbf{f}$ for several diffusion scales $s > 1$ (as in [45]), giving the GNN the ability to learn algorithms like the power method and more directly accessing spectral properties of the graph. The GNN model can be further generalized to replicate other operators on graphs. For instance, the pointwise nonlinearity $\eta$ can depend on the vertex type, allowing extremely rich architectures [50], [51], [55], [79], [80].

### Charting-based methods

We now consider the second subclass of non-Euclidean learning problems, where we are given multiple domains. A prototypical application the reader should have in mind throughout this section is the problem of finding correspondence between shapes, modeled as manifolds (see "Three-Dimensional Shape Correspondence Application"). As we have seen, defining convolution in the spectral domain has an inherent drawback of the inability to adapt the model across different domains. We will therefore need to resort to an alternative generalization of the convolution in the spatial domain that does not suffer from this drawback.

Furthermore, note that in the setting of multiple domains, there is no immediate way to define a meaningful spatial pooling operation, as the number of points on different domains can vary, and their order can be arbitrary. It is, however, possible to pool pointwise features produced by a network by aggregating all the local information into a single vector. One possibility for such a pooling is computing the statistics of the pointwise features, e.g., the mean or covariance [47]. Note that after such a pooling, all of the spatial information is lost.

On a Euclidean domain, due to shift invariance the convolution can be thought of as passing a template at each point of the domain and recording the correlation of the template with the function at that point. Thinking of image filtering, this amounts to extracting a (typically square) patch of pixels, multiplying it elementwise with a template and summing up the results, then moving to the next position in a sliding window manner. Shift invariance implies that the very operation of extracting the patch at each position is always the same.

One of the major problems in applying the same paradigm to non-Euclidean domains is the lack of shift invariance, implying that the patch operator extracting a local patch would be position dependent. Furthermore, the typical lack of meaningful global parameterization for a graph or manifold forces to represent the patch in some local intrinsic system of coordinates. Such a mapping can be obtained by defining a set of weighting functions $v_1(x,\cdot),\ldots,v_J(x,\cdot)$ localized to positions near $x$ (see examples in Figure 3). Extracting a patch amounts to averaging the function $f$ at each point by these weights,

$$D_j(x)f = \int_X f(x')\,v_j(x,x')\,dx', j = 1,\ldots,J, \tag{39}$$

providing for a spatial definition of an intrinsic equivalent of convolution

$$(f \star g)(x) = \sum_j g_j D_j(x) f, \tag{40}$$

where $g$ denotes the template coefficients applied on the patch extracted at each point. Overall, (39) and (40) act as a kind of nonlinear filtering of $f$, and the patch operator $D$ is specified by defining the weighting functions $v_1,\ldots,v_J$. Such filters are localized by construction, and the number of parameters is equal to the number of weighting functions $J = O(1)$. Several frameworks for non-Euclidean CNNs essentially amount to different choices of these weights. The spectrum-free methods (ChebNet and GCN) described in the previous section can also be thought of in terms of local weighting functions, as it is easy to see the analogy between (40) and (34).

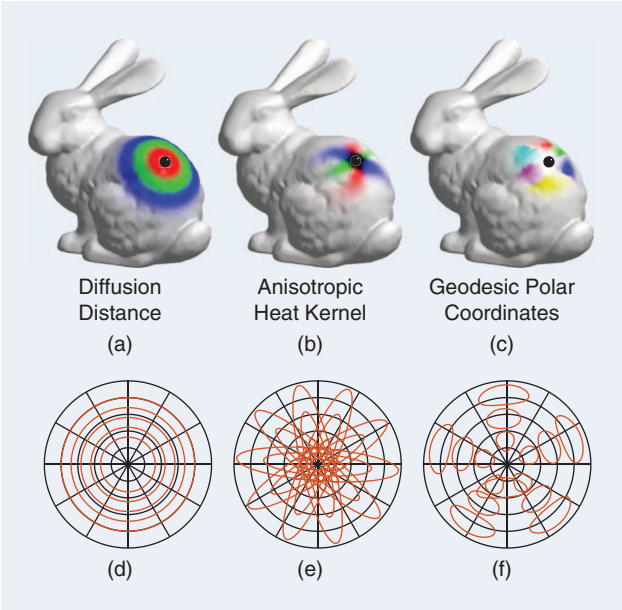



**FIGURE 3.** (a)–(c) The examples of intrinsic weighting functions used to construct a patch operator at the point marked in black (different colors represent different weighting functions). (a) Diffusion distance allows to map neighbor points according to their distance from the reference point, thus defining a 1-D system of local intrinsic coordinates. (b) Anisotropic heat kernels of different scale and orientations and (c) geodesic polar weights are 2-D systems of coordinates. (d)–(f) The representation of the weighting functions in the local polar $(\rho,\theta)$ system of coordinates (red curves represent the 0.5 level set).

## Geodesic CNN

Because manifolds naturally come with a low-dimensional tangent space associated with each point, it is natural to work in a local system of coordinates in the tangent space [47]. In particular, on 2-D manifolds one can create a polar system of coordinates around $x$ where the radial coordinate is given by some intrinsic distance $\rho(x') = d(x,x')$, and the angular coordinate $\theta(x)$ is obtained by ray shooting from a point at equispaced angles. The weighting functions in this case can be obtained as a product of Gaussians

$$v_{ij}(x,x') = e^{-(\rho(x')-\rho_i)^2/2\sigma_\rho^2} e^{-(\theta(x')-\theta_j)^2/2\sigma_\theta^2}, \tag{41}$$

where $i = 1,\ldots,J$ and $j = 1,\ldots,J'$ denote the indices of the radial and angular bins, respectively. The resulting $JJ'$ weights are bins of width $\sigma_\rho \times \sigma_\theta$ in the polar coordinates [Figure 3(c) and (f)].

## Anisotropic CNN

We have already seen the non-Euclidean heat equation (S5), whose heat kernel $h_t(x,\cdot)$ produces localized blob-like weights around the point $x$ [see Figure S3(a)]. Varying the diffusion time $t$ controls the spread of the kernel. However, such kernels are isotropic, meaning that the heat flows equally fast in all the directions. A more general anisotropic diffusion [48] equation on a manifold

$$f_t(x,t) = -\mathrm{div}\,(\mathbf{A}(x)\nabla f(x,t)) \tag{42}$$

involves the thermal conductivity tensor $\mathbf{A}(x)$ (in the case of 2-D manifolds, a $2 \times 2$ matrix is applied to the intrinsic gradient

in the tangent plane at each point), allowing modeling heat flow that is position and direction dependent [82]. A particular choice of the heat conductivity tensor proposed in [53] is

$$\mathbf{A}_{\alpha\theta}(x) = \mathbf{R}_\theta(x)\begin{pmatrix}\alpha & \\ & 1\end{pmatrix}\mathbf{R}_\theta^\top(x), \tag{43}$$

where the $2 \times 2$ matrix $\mathbf{R}_\theta(x)$ performs rotation of $\theta$ with respect to some reference (e.g., the maximum curvature) direction and $\alpha > 0$ is a parameter controlling the degree of anisotropy ($\alpha = 1$ corresponds to the classical isotropic case). The heat kernel of such anisotropic diffusion equation is given by the spectral expansion

$$h_{\alpha\theta t}(x,x') = \sum_{i\geq 0} e^{-t\lambda_{\alpha\theta i}}\phi_{\alpha\theta i}(x)\phi_{\alpha\theta i}(x'), \tag{44}$$

where $\phi_{\alpha\theta 0}(x), \phi_{\alpha\theta 1}(x), \ldots$ are the eigenfunctions and $\lambda_{\alpha\theta 0}, \lambda_{\alpha\theta 1}, \ldots$ the corresponding eigenvalues of the anisotropic Laplacian

$$\Delta_{\alpha\theta} f(x) = -\operatorname{div}(\mathbf{A}_{\alpha\theta}(x)\nabla f(x)). \tag{45}$$

The discretization of the anisotropic Laplacian is a modification of the cotangent formula (S12) on meshes or graph Laplacian (S9) on point clouds [48]. The anisotropic heat kernels $h_{\alpha\theta t}(x,\cdot)$ look like elongated rotated blobs [see Figure 3(b) and (e)], where the parameters $\alpha, \theta$ and $t$ control the elongation, orientation, and scale, respectively. Using such kernels as weighting functions $v$ in the construction of the patch operator (39), it is possible to obtain a charting similar to the geodesic patches (roughly, $\theta$ plays the role of the angular coordinate and $t$ of the radial one).

### *Mixture model network*

Finally, as the most general construction of patches, Monti et al. [54] proposed defining at each point a local system of $d$-dimensional pseudocoordinates $\mathbf{u}(x,x')$ around $x$. On these coordinates, a set of parametric kernels $v_1(\mathbf{u}), \ldots, v_J(\mathbf{u})$ is applied, producing the weighting functions in (39). Rather than using fixed kernels, as in the previous constructions, Monti et al. use Gaussian kernels

$$v_j(\mathbf{u}) = \exp\Big(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\mathbf{u}-\boldsymbol{\mu}_j)\Big),$$

whose parameters ($d \times d$ covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_J$ and $d \times 1$ mean vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_J$) are learned [this choice allows interpreting intrinsic convolution (40) as a mixture of Gaussians, hence the name of the approach]. Learning not only the filters but also the patch operators in (40) affords additional DoF to the mixture model network (MoNet) architecture, which makes it currently the state-of-the-art approach in several applications. It is also easy to see that this approach generalizes the previous models, and, e.g., classical Euclidean CNNs as well as geodesic and anisotropic CNNs can be obtained as particular instances thereof [54]. MoNet can also be applied on general graphs using as the pseudocoordinates $\mathbf{u}$ some local graph features, such as vertex degree, geodesic distance, and so forth.

## Combined spatial/spectral methods

The third alternative for constructing convolutionlike operations of non-Euclidean domains is jointly in spatial-frequency domain.

### *Windowed Fourier transform*

One of the notable drawbacks of classical Fourier analysis is its lack of spatial localization. By virtue of the uncertainty principle, one of the fundamental properties of Fourier transforms, spatial localization comes at the expense of frequency localization and vice versa. In classical signal processing, this problem is remedied by localizing frequency analysis in a window $g(x)$, leading to the definition of the windowed Fourier transform (WFT, also known as *short-time Fourier transform* or *spectrogram* in signal processing),

$$(Sf)(x,\omega) = \int_{-\infty}^{\infty} f(x')\underbrace{g(x'-x)e^{-i\omega x'}}_{\overline{g_{x,\omega}(x')}}dx' \tag{46}$$

$$= \langle f, g_{x,\omega}\rangle_{L^2(\mathbb{R})}. \tag{47}$$

The WFT is a function of two variables: spatial location of the window $x$ and the modulation frequency $\omega$. The choice of the window function $g$ allows control of the tradeoff between spatial and frequency localization (wider windows result in better frequency resolution). Note that WFT can be interpreted as inner products (47) of the function $f$ with translated and modulated windows $g_{x,\omega}$, referred to as the WFT *atoms*.

The generalization of such a construction to non-Euclidean domains requires the definition of translation and modulation operators [83]. While modulation simply amounts to multiplication by a Laplacian eigenfunction, translation is not well defined due to the lack of shift invariance. It is possible to resort again to the spectral definition of a convolution-like operation (26), defining translation as convolution with a delta function,

$$(g \star \delta_{x'})(x) = \sum_{i\geq 0}\langle g,\phi_i\rangle_{L^2(\mathcal{X})}\langle \delta_{x'},\phi_i\rangle_{L^2(\mathcal{X})}\phi_i(x)$$

$$= \sum_{i\geq 0}\hat{g}_i\phi_i(x')\phi_i(x). \tag{48}$$

The translated and modulated atoms can be expressed as

$$g_{x',j}(x) = \phi_j(x')\sum_{i\geq 0}\hat{g}_i\phi_i(x)\phi_i(x'), \tag{49}$$

where the window is specified in the spectral domain by its Fourier coefficients $\hat{g}$. The WFT on non-Euclidean domains thus takes the form

$$(Sf)(x',j) = \langle f, g_{x',j}\rangle_{L^2(\mathcal{X})} = \sum_{i\geq 0}\hat{g}_i\phi_i(x')\langle f,\phi_i\phi_j\rangle_{L^2(\mathcal{X})}. \tag{50}$$

Due to the intrinsic nature of all the quantities involved in its definition, the WFT is also intrinsic.

### Wavelets

Replacing the notion of frequency in time–frequency representations by that of scale leads to wavelet decompositions. Wavelets have been extensively studied in general graph domains [84]. Their objective is to define stable linear decompositions with atoms well localized both in space and frequency that can efficiently approximate signals with isolated singularities. Similarly to the Euclidean setting, wavelet families can be constructed either from spectral constraints or from spatial constraints.

The simplest of such families are Haar wavelets. Several bottom-up wavelet constructions on graphs were studied in [85] and [86]. In [87], the authors developed an unsupervised method that learns wavelet decompositions on graphs by optimizing a sparse reconstruction objective. In [88], ensembles of Haar wavelet decompositions were used to define deep wavelet scattering transforms on general domains, obtaining excellent numerical performance. Learning amounts to finding optimal pairings of nodes at each scale, which can be efficiently solved in polynomial time.

### Localized SCNN

Boscaini et al. used the WFT as a way of constructing patch operators (39) on manifolds and point clouds and used in an intrinsic convolution-like construction (40). The WFT allows expressing a function around a point in the spectral domain in the form $D_j(x)f = (Sf)(x,j)$ [89]. Applying learnable filters to such patches (which in this case can be interpreted as spectral multipliers), it is possible to extract meaningful features that also appear to generalize across different domains. An additional DoF is the definition of the window, which can also be learned [89].

## Applications

### Network analysis

One of the classical examples used in many works on network analysis is citation networks. A citation network is a graph where vertices represent articles and there is a directed edge $(i, j)$ if article $i$ cites article $j$. Typically, vertex-wise features representing the content of the article (e.g., histogram of frequent terms in the article) are available. A prototypical classification application is to attribute each article to a field. Traditional approaches work vertex-wise, performing classification of each vertex's feature vector individually. More recently, it was shown that classification can be considerably improved using information from neighbor vertices, e.g., with a CNN on graphs [45], [77]. An example of the application of spectral and spatial graph CNN models on a citation network is shown in "Citation Network Analysis Application."

Another fundamental problem in network analysis is ranking and community detection. These can be estimated by solving an eigenvalue problem on an appropriately defined operator on the graph. For instance, the Fiedler vector (the eigenvector associated with the smallest nontrivial eigenvalue of the Laplacian) carries information on the graph partition with minimal cut [73], and the popular PageRank algorithm approximates page ranks with the principal eigenvector of a modified Laplacian operator. In some contexts, one may want develop data-driven versions of such algorithms that can adapt to model mismatch and perhaps provide a faster alternative to diagonalization methods. By unrolling power iterations, one obtains a GNN architecture whose parameters can be learned with backpropagation from labeled examples, similarly to the learned sparse coding paradigm [91]. We are currently exploring this connection by constructing multiscale versions of GNNs.

### Recommender systems

Recommending movies on Netflix, friends on Facebook, or products on Amazon are a few examples of recommender systems that have recently become ubiquitous in a broad range of applications. Mathematically, a recommendation method can be posed as a matrix completion problem [92], where columns and rows represent users and items, respectively, and matrix values represent a score determining whether a user would like an item or not. Given a small subset of known elements of the matrix, the goal is to fill in the rest. A famous example is the Netflix challenge [93] offered in 2009 and carrying a US$1 million prize for the algorithm that can best predict user ratings for movies based on previous ratings. The size of the Netflix matrix is 480,000 movies × 18,000 users (8.5 billion elements), with only 0.011% known entries.

**Recommending movies on Netflix, friends on Facebook, or products on Amazon are a few examples of recommender systems that have recently become ubiquitous in a broad range of applications. Such applications may benefit from geometric deep learning methods.**

Several recent works proposed to incorporate geometric structure into matrix completion problems [94]–[97] in the form of column and row graphs representing similarity of users and items, respectively (see Figure 4). Such a geometric matrix completion setting makes meaningful, e.g., the notion of smoothness of the matrix values and was shown beneficial for the performance of recommender systems.

In a recent work, Monti et al. [56] proposed addressing the geometric matrix completion problem by means of a learnable model combining a multigraph CNN (MGCNN) and a recurrent neural network (RNN). Multigraph convolution can be thought of as a generalization of the standard bidimensional image convolution, where the domains of the rows and the columns are now different (in our case, user and item graphs). The features extracted from the score matrix by means of the MGCNN are then passed to an RNN, which produces a sequence of incremental updates of the score values. Overall, the model can be considered as a learnable diffusion of the scores, with the main advantage compared to traditional approach being a fixed number of variables independent of the matrix size. The MGCNN achieved

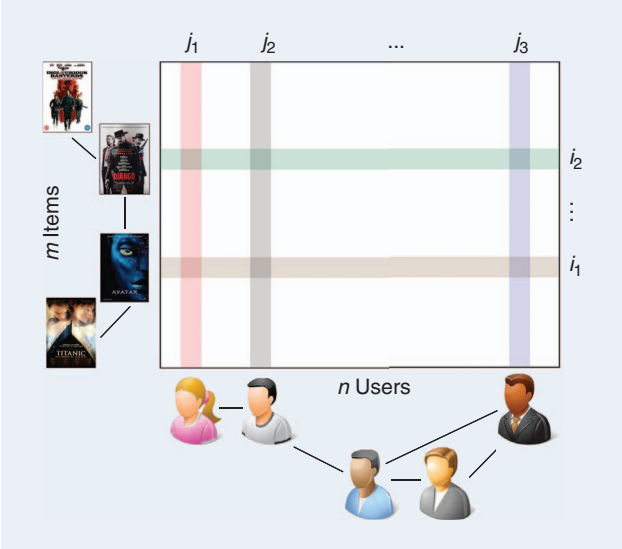



**FIGURE 4.** The geometric matrix completion exemplified on the famous Netflix movie recommendation problem. The column and row graphs represent the relationships between users and items, respectively.

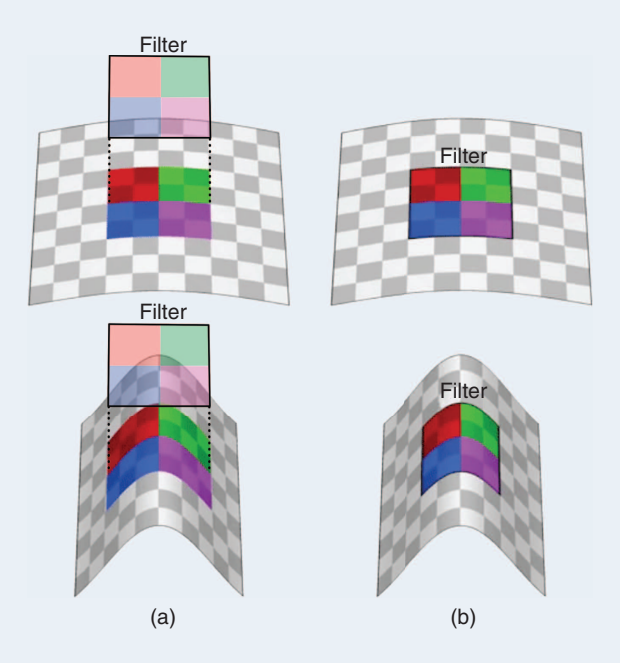



**FIGURE 5.** An illustration of the difference between (a) classical CNN applied to a 3-D shape (checkered surface) considered as a Euclidean object and (b) a geometric CNN applied intrinsically on the surface. In the latter case, the convolutional filters (visualized as a colored window) are deformation invariant by construction.

state-of-the-art results on several classical matrix completion challenges and, on a more conceptual level, could be a very interesting practical application of geometric deep learning to a classical signal processing problem of matrix completion.

### *Computer vision and graphics*

The computer-vision community has recently shown an increasing interest in working with 3-D geometric data, mainly due to the emergence of affordable range-sensing technology, such as Microsoft Kinect or Intel RealSense. Many machine-learning techniques successfully working on images were tried as is on 3-D geometric data, represented for this purpose in some way digestible by standard frameworks, e.g., as range images [98], [99] or rasterized volumes [100], [101]. The main drawback of such approaches is their treatment of geometric data as Euclidean structures. First, for complex 3-D objects, Euclidean representations, such as depth images or voxels, may lose significant parts of the object or its fine details or even break its topological structure. Second, Euclidean representations are not intrinsic and vary when changing pose or deforming the object. Achieving invariance to shape deformations, a common requirement in many vision applications, demands very complex models and huge training sets due to the large number of DoF involved in describing nonrigid deformations [see Figure 5(a)].

**The computer-vision community has recently shown an increasing interest in working with 3-D geometric data.**

In the domain of computer graphics, on the other hand, working intrinsically with geometric shapes is a standard practice. In this field, 3-D shapes are typically modeled as Riemannian manifolds and are discretized as meshes. Numerous studies (see, e.g., [102]–[106]) have been devoted to designing local and global features, e.g., for establishing similarity or correspondence between deformable shapes with guaranteed invariance to isometries.

However, different applications in computer vision and graphics may require completely different features. For instance, to establish feature-based correspondence between a collection of human shapes, one would desire the descriptors of corresponding anatomical parts (e.g., noses, mouths) to be as similar as possible across the collection (see Figure 6(a)). In other words, such descriptors should be invariant to the collection variability. Conversely, for shape classification, one would like descriptors that emphasize the subject-specific characteristics and, e.g., distinguish between two different nose shapes (see Figure 6b). Deciding a priori which structures should be used and which should be ignored is often hard or sometimes even impossible. Moreover, axiomatic modeling of geometric noise, such as 3-D scanning artifacts, turns out to be extremely hard.

By resorting to intrinsic deep neural networks on manifolds, the invariance to isometric deformations is automatically built into the model, thus vastly reducing the number of DoF required to describe the invariance class. Roughly speaking, the intrinsic deep model will try to learn residual deformations that deviate from the isometric model. Geometric deep learning can be applied to several problems in 3-D shape analysis, which can be divided into two classes. First are problems like local descriptor learning [47], [53] or correspondence learning [48] (see the example in "Three-Dimensional

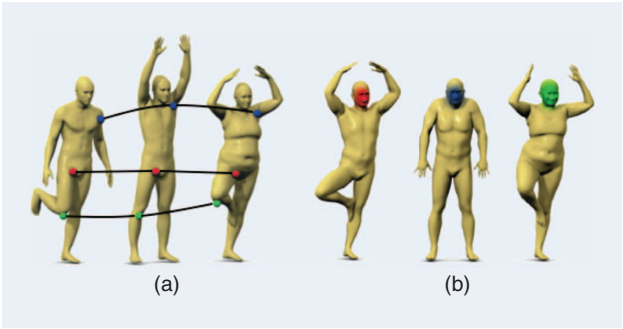



**FIGURE 6.** (a) The features used for shape correspondence should ideally manifest invariance across the shape class (e.g., the knee feature shown here should not depend on the specific person). (b) The features used for shape retrieval, on the contrary, should be specific to a shape within the class to allow distinguishing between different people. Similar features are marked with the same color. Handcrafting the right feature for each application is a very challenging task.

Shape Correspondence Application"), in which the output of the network is pointwise. The inputs to the network are some pointwise features, e.g., color texture or simple geometric features, such as normals. Using a CNN architecture with multiple intrinsic convolutional layers, it is possible to produce nonlocal features that capture the context around each point. The second type of problems, such as shape recognition, require the network to produce a global shape descriptor, aggregating all the local information into a single vector using, e.g., the covariance pooling [47].

**The recent emergence of geometric deep-learning methods in various communities and application domains allows us to proclaim that we might be witnessing a new field being born.**

### *Particle physics and chemistry*

Many areas of experimental science are interested in studying systems of discrete particles defined over a low-dimensional phase space. For instance, the chemical properties of a molecule are determined by the relative positions of its atoms, and the classification of events in particle accelerators depends upon position, momentum, and spin of all the particles involved in the collision.

The behavior of an $N$-particle system is ultimately derived from solutions of the Schrödinger equation, but its exact solution involves diagonalizing a linear system of exponential size. In this context, an important question is whether one can approximate the dynamics with a tractable model that incorporates by construction the geometric stability postulated by the Schrödinger equation and at the same time has enough flexibility to adapt to data-driven scenarios and capture complex interactions.

An instance $l$ of an $N_l$-particle system can be expressed as

$$f_l(t) = \sum_{j=1}^{N_l} \alpha_{j,l}\delta(t - x_{j,l}),$$

where $(\alpha_{j,l})$ model particle-specific information, such as the spin, and $(x_{j,l})$ are the locations of the particles in a given phase space. Such a system can be recast as a signal defined over a graph with $|\mathcal{V}_l| = N_l$ vertices and edge weights $\mathbf{W}_l = (\phi(\alpha_{i,l}, \alpha_{j,l}, x_{i,l}, x_{j,l}))$ expressed through a similarity kernel capturing the appropriate priors. GNNs are currently being applied to perform event classification, energy regression, and anomaly detection in high-energy physics experiments, such as the Large Hadron Collider, and neutrino detection in the IceCube Observatory. Recently, models based on GNNs have been applied to predict the dynamics of $N$-body systems [111], [112], showing excellent prediction performance.

### *Molecule design*

A key problem in material and drug design is predicting the physical, chemical, or biological properties (such as solubility of toxicity) of a novel molecule from its structure. State-of-the-art methods rely on hand-crafted molecule descriptors, such as circular fingerprints [113]–[115]. A recent work from Harvard University in Cambridge, Massachusetts [55] proposed modeling molecules as graphs (where vertices represent atoms and edges represent chemical bonds) and employing GCNNs to learn the desired molecule properties. The authors' approach has significantly outperformed handcrafted features. This work opens a new avenue in molecule design that might revolutionize the field.

### *Medical imaging*

An application area where signals are naturally collected on non-Euclidean domains and where the methodologies we reviewed could be very useful is brain imaging. A recent trend in neuroscience is to associate functional magnetic resonance imaging traces with a precomputed connectivity rather than inferring it from the traces themselves [116]. In this case, the challenge consists in processing and analyzing an array of signals collected over a complex topology, which results in subtle dependencies. For example, in a recent work from Imperial College London [117], GCNNs were used to detect disruptions of the brain functional networks associated with autism.

## Open problems and future directions

The recent emergence of geometric deep-learning methods in various communities and application domains, which we tried to overview in this article, allows us to proclaim, perhaps with some caution, that we might be witnessing a new field being born. We expect the following years to bring exciting new methods and applications, and conclude our review with a few observations of current key difficulties and potential directions of future research.

Many disciplines dealing with geometric data employ some empirical models or handcrafted features. This is a typical situation in geometry processing and computer graphics, where axiomatically constructed features are used to analyze 3-D shapes, or computational sociology, where it is common

to first come up with a hypothesis and then test it on the data [22]. Yet, such models assume some prior knowledge (e.g., isometric shape deformation model) and often fail to correctly capture the full complexity and richness of the data. In computer vision, departing from handcrafted features toward generic models learnable from the data in a task-specific manner has brought a breakthrough in performance and led to an overwhelming trend in the community to favor deep-learning methods. Such a shift has not occurred yet in the fields dealing with geometric data due to the lack of adequate methods, but there are the first indications of a coming paradigm shift.

### Generalization

Generalizing deep-learning models to geometric data requires not only finding non-Euclidean counterparts of basic building blocks (such as convolutional and pooling layers) but also generalization across different domains. Generalization capability is a key requirement in many applications, including computer graphics, where a model is learned on a training set of non-Euclidean domains (3-D shapes) and then applied to previously unseen ones. Spectral formulation of convolution allows designing CNNs on a graph, but the model learned this way on one graph cannot be straightforwardly applied to another one, because the spectral representation of convolution is domain dependent. A possible remedy to the generalization problem of spectral methods is the recent architecture proposed in [118], applying the idea of spatial transformer networks [119] in the spectral domain. This approach is reminiscent of the construction of compatible orthogonal bases by means of joint Laplacian diagonalization [75], which can be interpreted as an alignment of two Laplacian eigenbases in a $k$-dimensional space.

The spatial methods, on the other hand, allow generalization across different domains, but the construction of low-dimensional local spatial coordinates on graphs turns out to be rather challenging. In particular, the construction of anisotropic diffusion on general graphs is an interesting research direction. The spectrum-free approaches also allow generalization across graphs, at least in terms of their functional form. However, if multiple layers of (38) are used with no nonlinearity or learned parameters $\theta$, simulating a high power of the diffusion, the model may behave differently on different kinds of graphs. Understanding under what circumstances and to what extent these methods generalize across graphs is currently being studied.

### Time-varying domains

An interesting extension of geometric deep-learning problems discussed in this review is coping with signals defined over a dynamically changing structure. In this case, we cannot assume a fixed domain and must track how these changes affect signals. This could prove useful to tackle applications like abnormal activity detection in social or financial networks. In the domain of computer graphics and vision, potential applications deal with dynamic shapes (e.g., 3-D video captured by a range sensor).

### Directed graphs

Dealing with directed graphs is also a challenging topic, as such graphs typically have nonsymmetric Laplacian matrices that do not have orthogonal eigendecomposition allowing easily interpretable spectral-domain constructions. Citation networks, which are directed graphs, are often treated as undirected graphs (including in our example in "Three-Dimensional Shape Correspondence Application") considering citations between two articles without distinguishing which article cites which. This obviously may lose important information.

### Synthesis problems

Our main focus in this review was primarily on analysis problems on non-Euclidean domains. Not less important is the question of data synthesis. There have been several recent attempts to try to learn a generative model allowing to synthesize new images [120] and speech waveforms [121]. Extending such methods to the geometric setting seems a promising direction, though the key difficulty is the need to reconstruct the geometric structure (e.g., an embedding of a 2-D manifold in the 3-D Euclidean space modeling a deformable shape) from some intrinsic representation [122].

**One of the main reasons for the computational efficiency of deep-learning architectures is the assumption of regularly structured data on a 1-D or 2-D grid.**

### Computation

The final consideration is a computational one. All existing deep-learning software frameworks are primarily optimized for Euclidean data. One of the main reasons for the computational efficiency of deep-learning architectures (and one of the factors that contributed to their renaissance) is the assumption of regularly structured data on a 1-D or 2-D grid, allowing to take advantage of modern GPU hardware. Geometric data, on the other hand, in most cases do not have a grid structure, requiring different ways to achieve efficient computations. It seems that computational paradigms developed for large-scale graph processing are more adequate frameworks for such applications.

## Acknowledgments

We are grateful to Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Xavier Bresson, Thomas Kipf, and Michaël Defferard for their comments and for providing some of the figures used in this article. This work was supported in part by the European Research Council grant numbers 307047 (COMET) and 724228 (LEMAN), a Google Faculty Research Award, a Radcliffe fellowship, a Rudolf Diesel fellowship, and Nvidia equipment grants.

## Authors

***Michael M. Bronstein*** (michael.bronstein@usi.ch) received his B.S. degree in electrical engineering and his Ph.D. degree in computer science from the Technion Israel Institute of Technology, Haifa, in 2002 and 2007, respectively. He is currently an associate professor at the University of Lugano, Switzerland, and Tel Aviv University, Israel. He also serves as a principal engineer at Intel Perceptual Computing, Israel. His main research interest is in theoretical and computational methods for geometric data analysis. He was selected as an ACM Distinguished Speaker, a World Economic Forum Young Scientist, and a member of the Young Academy of Europe. He received three ERC grants, a Radcliffe fellowship from Harvard University, a Rudolf Diesel fellowship from the Technical University of Munich, and a Google faculty award.

***Joan Bruna*** (bruna@cims.nyu.edu) received his B.S. degree in mathematics and electrical engineering in 2002 from the Universitat Politècnica de Catalunya, Barcelona, Spain, his M.S. degree in applied mathematics in 2005 from the École normale supérieure Cachan, France, and his Ph.D. degree in applied mathematics in 2013 from the École Polytechnique, Palaiseau, France. He was a postdoctoral researcher at the Courant Institute, New York University (NYU), and a postdoctoral fellow at Facebook AI Research, Menlo Park, California. In 2015, he became an assistant professor at the University of California, Berkeley Statistics Department, and, starting in the fall of 2016, he joined the Courant Institute, NYU, as an assistant professor in computer science, data science, and mathematics (affiliated). His research interests include invariant signal representations, high-dimensional statistics and stochastic processes, and deep learning and its applications to signal processing.

***Yann LeCun*** (yann@fb.com) received his electrical engineering diploma in 1983 from the Ecole Superieure d'Ingenieurs en Electrotechnique et Electronique, Paris, and his Ph.D. degree in computer science from the Universitè Pierre et Marie Curie, Paris, France, in 1987. After postdoctoral research at the University of Toronto, he joined AT&T Bell Labs in Holmdel, New Jersey, in 1988, where he became head of the Image Processing Research Department. He joined New York University (NYU) as a professor in 2003 after a brief period as a fellow of the NEC Research Institute in Princeton, New Jersey. He is currently the director of artificial intelligence research at Facebook and a professor at NYU. Since the late 1980s, he has been working on deep-learning methods, particularly the convolutional network model. He has been on the editorial board of *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *IEEE Transactions on Neural Networks*. He received the IEEE Neural Network Pioneer Award in 2014.

***Arthur Szlam*** (aszlam@fb.com) received his Ph.D. degree in computer science from Yale University, New Haven, Connecticut. He is a research scientist at Facebook AI Research, Menlo Park, California. Prior to joining Facebook, he was a postdoctoral fellow at the Institute for Mathematics and Its Applications at the University of Minnesota, Minneapolis, and the Institute for Pure and Applied Mathematics, University of California, Los Angeles, and an assistant professor at City College of New York. He is a Sloan Foundation fellow. His research interests are in computational harmonic analysis, the relationships between smoothness, frequency, and scale on graphs and data clouds, and applications to signal processing and machine learning.

***Pierre Vandergheynst*** (pierre.vandergheynst@epfl.ch) received his M.S. degree in theoretical physics in 1994 and his Ph.D. degree in mathematical physics from the Université catholique de Louvain, Belgium, in 1998. He is a full professor at the Ecole Polytechnique Federale de Lausanne, Switzerland. He has authored or coauthored more than 50 journal papers, one monograph, and several book chapters. He has been an associate editor of *IEEE Transactions on Signal Processing* since 2007, a member of technical committees for various conferences, and general cochair of the 2008 European Signal Processing Conference. He is a laureate of the Apple Research and Technology Support Award and of the 2010–2011 De Boelpaepe prize from the Royal Academy of Sciences of Belgium. He is strongly involved in technology transfer, having cofounded two startups, and holds numerous patents. His research focuses on harmonic analysis, sparse approximations, and mathematical image processing with applications to higher-dimensional, complex data processing.

## References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *Proc. 2011 IEEE Workshop Automatic Speech Recognition and Understanding*, pp. 196–201.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2014, pp. 3104–3112.

[5] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2010, pp. 253–256.

[6] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, 2011, pp. 253–256.

[7] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[9] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1701–1708.

[10] K. Simonyan and A. Zisserman. (2014). Very deep convolutional networks for large-scale image recognition. [Online]. Available: https://arXiv:1409.1556

[11] K. He, X. Zhang, S. Ren, and J. Sun. (2015). Deep residual learning for image recognition. [Online]. Available: https://arXiv:1512.03385

[12] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[14] E. P. Simoncelli and B. A. Olshausen, "Natural image statistics and neural representation," *Ann. Rev. Neuroscience*, vol. 24, no. 1, pp. 1193–1216, 2001.

[15] D. J. Field, "What the statistics of natural images tell us about visual coding," in *Proc. SPIE-Int. Society for Optical Engineering*, vol. 1077, p. 269, 1989.

[16] P. Mehta and D. J. Schwab. (2014). An exact mapping between the variational renormalization group and deep learning. [Online]. Available: https://arXiv:1410.3831

[17] S. Mallat, "Group invariant scattering," *Commun. Pure and Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, 2012.

[18] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, 2013.

[19] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, and A. Szlam, "A mathematical motivation for complex-valued convolutional networks," *Neural Computation*, vol. 28, no. 5, pp. 815–825, 2016.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten ZIP code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[21] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. (2013). Maxout networks. [Online]. Available: https://arXiv:1302.4389

[22] D. Lazer, A. Pentland, L. Adamic, S. Aral, A. L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. Van Alstyne, "Life in the network: The coming age of computational social science," *Science*, vol. 323, no. 5915, pp. 721–723, 2009.

[23] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C. H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, O. Otim, C. T. Brown, C. B. Livi, P. Y. Lee, R. Revilla, A. G. Rust, Z. Pan, M. J. Schilstra, P. J. Clarke, M. I. Arnone, L. Rowen, R. A. Cameron, D. R. McClay, L. Hood, and H. Bolouri, "A genomic regulatory network for development," *Science*, vol. 295, no. 5560, pp. 1669–1678, 2002.

[24] M. B. Wakin, D. L. Donoho, H. Choi, and R. G. Baraniuk, "The multiscale structure of non-differentiable image manifolds," in *Proc. SPIE-Int. Society for Optical Engineering*, vol. 5914, p. 59141B, 2005.

[25] N. Verma, S. Kpotufe, and S. Dasgupta, "Which spatial partition trees are adaptive to intrinsic dimension?" in *Proc. Uncertainty in Artificial Intelligence*, 2009, pp. 565–574.

[26] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[27] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[28] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *JMLR*, vol. 9, pp. 2579–2605, 2008.

[29] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[30] R. R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmon. Anal.*, vol. 21, no. 1, pp. 5–30, 2006.

[31] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 1735–1742.

[32] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2014, pp. 701–710.

[33] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. Int. World Wide Web Conf. (WWW)*, 2015, pp. 1067–1077.

[34] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. Int. Conf. Information and Knowledge Management (CIKM)*, 2015, pp. 891–900.

[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). Efficient estimation of word representations in vector space. [Online]. Available: https://arXiv:1301.3781

[36] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[37] N. Pržulj, "Biological network comparison using graphlet degree distribution," *Bioinformatics*, vol. 23, no. 2, pp. 177–183, 2007.

[38] J. Sun, M. Ovsjanikov, and L. J. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Comput. Graph. Forum*, vol. 28, no. 5, pp. 1383–1392, 2009.

[39] R. Litman and A. M. Bronstein, "Learning spectral descriptors for deformable shape correspondence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 171–180, 2014.

[40] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, no. 3, pp. 75–174, 2010.

[41] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2013, pp. 3111–3119.

[42] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2011, pp.1082–1090.

[43] 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 Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[44] M. Henaff, J. Bruna, and Y. LeCun. (2015). Deep convolutional networks on graph-structured data. [Online]. Available: https://arXiv:1506.05163

[45] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2016, pp. 3844–3852.

[46] J. Atwood and D. Towsley. (2016). Diffusion-convolutional neural networks. [Online]. Available: https://arXiv:1511.02136v2

[47] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds," in *Proc. Int. IEEE Workshop 3-D Representation and Recognition (3DRR)*, 2015, pp. 832–840.

[48] D. Boscaini, J. Masci, E. Rodolà, and M. M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2016, pp. 3189–3197.

[49] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, 2005, pp. 729–734.

[50] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. (2015). Gated graph sequence neural networks. [Online]. Available: https://arXiv:1511.05493

[51] S. Sukhbaatar, A. Szlam, and R. Fergus. (2016). Learning multiagent communication with backpropagation. [Online]. Available: https://arXiv:1605.07736

[52] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. (2013). Spectral networks and locally connected networks on graphs. [Online.] Available: https://arXiv.1312.6203

[53] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers, "Anisotropic diffusion descriptors," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 431–441, 2016.

[54] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, (2017). Geometric deep learning on graphs and manifolds using mixture model CNNs. [Online.] Available: https://arXiv.1611.08402

[55] 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. Int. Conf. Neural Information Processing Systems (NIPS)*, 2015, pp. 2224–2232.

[56] F. Monti, X. Bresson, and M. M. Bronstein. (2017). Geometric matrix completion with recurrent multi-graph neural networks. [Online]. Available: https://arXiv.1704.06803

[57] S. Mallat, "Understanding deep convolutional networks," *Philos. Trans. R. Soc. London A, Math. Phys. Sci.*, vol. 374, no. 2065, 2016. doi: 10.1098/rsta.2015.0207.

[58] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010.

[59] A. Dosovitskiy, P. Fischery, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox "Flownet: Learning optical flow with convolutional networks," in *Proc. 2015 Int. Conf. Computer Vision*, 2015, pp. 2758–2766.

[60] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (2014). Striving for simplicity: The all convolutional net. [Online]. Available: https://arXiv:1412.6806

[61] S. Mallat, *A Wavelet Tour of Signal Processing*. New York: Academic, 1999.

[62] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. 18th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2015, pp. 192–204.

[63] I. Safran and O. Shamir. (2015). On the quality of the initial basin in overspecified neural networks. [Online]. Available: https://arXiv:1511.04210

[64] K. Kawaguchi, "Deep learning without poor local minima," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2016, pp. 586–594.

[65] T. Chen, I. Goodfellow, and J. Shlens. (2015). Net2net: Accelerating learning via knowledge transfer. [Online]. Available: https://arXiv:1511.05641

[66] C. D. Freeman and J. Bruna. (2017). Topology and geometry of half-rectified network optimization. [Online]. Available: https://arXiv.1611.01540

[67] J. Nash, "The imbedding problem for Riemannian manifolds," *Ann. Math.*, vol. 63, no. 1, pp. 20–63, 1956.

[68] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun, "Discrete Laplace operators: No free lunch," in *Proc. 5th Eurographics Symp. Geometry Processing (SGP)*, 2007, pp. 33–37.

[69] M. Wardetzky, "Convergence of the cotangent formula: An overview," in *Discrete Differential Geometry*. Cambridge, MA: Birkhäuser, 2008, pp. 275–286.

[70] U. Pinkall and K. Polthier, "Computing discrete minimal surfaces and their conjugates," *Exp. Mathematics*, vol. 2, no. 1, pp. 15–36, 1993.

[71] S. Rosenberg, *The Laplacian on a Riemannian Manifold: An Introduction to Analysis on Manifolds*. Cambridge, U.K.: Cambridge Univ. Press, 1997.

[72] L.-H. Lim. (2015). Hodge Laplacians on graphs. [Online]. Available: https://arXiv:1507.05379

[73] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[74] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel, "Coupled quasi-harmonic bases," *Comput. Graph. Forum*, vol. 32, no. 2, pp. 439–448, 2013.

[75] D. Eynard, A. Kovnatsky, M. M. Bronstein, K. Glashoff, and A. M. Bronstein, "Multimodal manifold analysis by simultaneous diagonalization of Laplacians," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 12, pp. 2505–2517, 2015.

[76] N. L. Roux, Y. Bengio, P. Lamblin, M. Joliveau, and B. Kégl, "Learning the 2-d topology of images," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2008, pp. 841–848.

[77] T. N. Kipf and M. Welling. (2016). Semi-supervised classification with graph convolutional networks. [Online]. Available: https://arXiv:1609.02907

[78] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, 2009.

[79] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. (2016). A compositional object-based approach to learning physical dynamics. [Online]. Available: https://arXiv:1612.00341

[80] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2016, pp. 4502–4510.

[81] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2011, pp. 2528–2536.

[82] M. Andreux, E. Rodolà, M. Aubry, and D. Cremers, "Anisotropic Laplace-Beltrami operators for shape analysis," in *Proc. 6th Workshop Non-Rigid Shape Analysis and Deformable Image Alignment (NORDIA)*, 2014, pp. 299–312.

[83] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Appl. Comput. Harmon. Anal.*, vol. 40, no. 2, pp. 260–291, 2016.

[84] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Appl. Comput. Harmonic Anal.*, vol. 21, no. 1, pp. 53–94, 2006.

[85] A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: Top-down and bottom-up constructions," *Proc. SPIE-Int. Society for Optical Engineering*, vol. 5914, p. 59141D, 2005.

[86] M. Gavish, B. Nadler, and R. R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning," in *Proc. 27th Int. Conf. Machine Learning (ICML-10)*, 2010, pp. 367–374.

[87] R. Rustamov and L. J. Guibas, "Wavelets on graphs via deep learning," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2013, pp. 998–1006.

[88] X. Cheng, X. Chen, and S. Mallat, "Deep Haar scattering networks," *Information and Inference*, vol. 5, pp. 105–133, 2016.

[89] 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," *Comput. Graph. Forum*, vol. 34, no. 5, pp. 13–23, 2015.

[90] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.

[91] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Machine Learning (ICML-10)*, 2010, pp. 399–406.

[92] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Commun. ACM*, vol. 55, no. 6, pp. 111–119, 2012.

[93] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[94] H. Ma, D. Zhou, C. Liu, M. Lyu, and I. King, "Recommender systems with social regularization," in *Proc. Web Search and Data Mining*, 2011, pp. 287–296.

[95] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. (2014). Matrix completion on graphs. [Online]. Available: https://arXiv:1408.1717

[96] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2015, pp. 2107–2115.

[97] D. Kuang, Z. Shi, S. Osher, and A. Bertozzi. (2016). A harmonic extension approach for collaborative ranking. [Online]. Available: https://arXiv:1602.05127

[98] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. Int. Conf. Computer Vision (ICCV)*, 2015, pp. 945–953.

[99] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, "Dense human body correspondences using convolutional networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1544–1553.

[100] 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. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[101] 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. 2016 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5648–5656.

[102] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 103, no. 5, pp. 1168–1172, 2006.

[103] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1704–1711.

[104] V. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," *ACM Trans. Graph.*, vol. 30, no. 4, p. 79, 2011.

[105] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, "ShapeGoogle: Geometric words and expressions for invariant shape retrieval," *ACM Trans. Graph.*, vol. 30, no. 1, p. 1, 2011.

[106] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. J. Guibas, "Functional maps: A flexible representation of maps between shapes," *ACM Trans. Graph.*, vol. 31, no. 4, p. 30, 2012.

[107] S. Biasotti, A. Cerri, A. M. Bronstein, and M. M. Bronstein, "Recent trends, applications, and perspectives in 3D shape similarity assessment," *Comput. Graph. Forum*, vol. 35, no. 6, pp. 87–119, 2016.

[108] E. Rodolà, S. Rota Bulo, T. Windheuser, M. Vestner, and D. Cremers, "Dense non-rigid shape correspondence using random forests," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 4177–4184.

[109] O. Litany, E. Rodolà, A. M. Bronstein, and M. M. Bronstein. (2017). Deep functional maps: Structured prediction for dense shape correspondence. [Online]. Available: https://arXiv.1704.08686

[110] F. Tombari, S. Salti, and L. D Stefano, "Unique signatures of histograms for local surface description," in *Proc. European Conf. Computer Vision (ECCV)*, 2010, pp. 356–369.

[111] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4502–4510.

[112] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. (2016). A compositional object-based approach to learning physical dynamics. [Online]. Available: https://arXiv.1612.00341

[113] H. L. Morgan, "The generation of a unique machine description for chemical structure," *J. Chem. Documentation*, vol. 5, no. 2, pp. 107–113, 1965.

[114] R. C. Glem, A. Bender, C. H. Arnby, L. Carlsson, S. Boyer, and J. Smith, "The generation of a unique machine description for chemical structure," *Investigational Drugs*, vol. 9, no. 3, pp. 199–204, 2006.

[115] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *J. Chem. Inform. and Modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[116] M. G. Preti, T. A. Bolton, and D. Van De Ville. (2016). The dynamic functional connectome: State-of-the-art and perspectives,. *Science*. [Online] http://www.sciencedirect.com/science/article/pii/S1053811916307881

[117] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert. (2017). Distance metric learning using graph convolutional networks: Application to functional brain networks. [Online]. Available: https://arXiv:1703.02161

[118] L. Yi, H. Su, X. Guo, and L. J. Guibas. (2017). SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. [Online]. Available: https://arXiv.1612.00606

[119] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Int. Conf. Neural Information Processing Systems (NIPS)*, 2015, pp. 2017–2025.

[120] A. Dosovitskiy, J. Springenberg, M. Tatarchenko, and T. Brox, "Learning to generate chairs, tables and cars with convolutional networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1538–1546.

[121] S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. (2016). Wavenet: A generative model for raw audio. [Online]. Available: https://arXiv:1609.03499

[122] D. Boscaini, D. Eynard, D. Kourounis, and M. M. Bronstein, "Shape-from-operator: Recovering shapes from intrinsic operators," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 265–274, 2015.

SP