Deep Learning on Non-Euclidean Domains

All learning settings we have encountered thus far had a common (sometimes, tacit) property: they assumed Euclidean geometry of the data. For example, we could compute standard inner products, subtract one vector from another, apply matrices to vectors, etc. Data like time signal and images were further discretized on regular Cartesian grids, and we could apply operations like convolution by simply sliding the same window over the signal and computing inner products.

Most of these apparently straightforward notions become less straightforward when the domain underlying the data is no longer Euclidean. Such kinds of data arise in a long list of applications. For example, in social networks, user information can be modeled as signals on a graph. Sensor networks are also modeled as graphs of distributed interconnected sensors, whose readings are time-dependent signals on the graph. In neuroscience, graph models are used to represent anatomical and functional structures of the brain. In biology, graphs are a common way to express interactions between genes, proteins, etc. In computer graphics and vision, three-dimensional geometric objects are often represented as Riemannian manifolds (surfaces) endowed with attributes such as color texture.

It is important to distinguish between two very distinct tasks: learning on non-Euclidean domains vs. learning of non-Euclidean domains. We encountered the latter problem when discussing unsupervised learning, where our goal was to discover (learn) the latent manifold from which the data are sampled. The former problem of analyzing signals on graphs and manifolds is what is going to occupy us in this lecture. In what follows, we will briefly review the main properties of the basic ingredients of CNNs on Euclidean domain, specifically, the convolution operator and pooling. We will then describe how to generalize these notions to graphs.

1 Convolution on Euclidean domains

The main ingredient of a CNN is a convolutional layer, describing a mapping between $m$-dimensional input signals to $n$-dimensional output signals (we use the term signal to generalize the notion of a sequence, allowing its elements to be indexed by a $d$-dimensional multi-index; this notion includes images and higher-dimensional signals besides time series). Formally, a convolutional layer accepts an $m$-dimensional vector-valued (infinitely supported) signal $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^m) = \{(x^1_k, \ldots, x^m_k)\}_{k \in \mathbb{Z}^d}$, each input dimension of which is called a channel or feature map. The layer produces an $n$-dimensional (infinitely supported) signal $\mathbf{y} = (\mathbf{y}^1, \ldots, \mathbf{y}^n) = \{(y^1_k, \ldots, y^n_k)\}_{k \in \mathbb{Z}^d}$ by applying a bank of filters,

$$
\mathbf{y}^j = \varphi \left( \sum_{i=1}^{m} w^{ij} \ast \mathbf{x}^i + b_j \right),
$$

where $\ast$ denotes convolution.
Explicitly, the action of the convolution $z^j = w^{ij} \ast x^i$ can be written as

$$z^j_k = \sum_{i=1}^{m} \sum_{p \in \mathbb{Z}^d} w^{ij}_{pi} x_i^{p-k}.$$ 

Note the $d$-dimensional multi-indices in the sum.

### 1.1 Eigenvectors of Toeplitz operators

Since the convolution operation is the main ingredient of a CNN, let us dedicate some attention to listing a few of its properties that will be instrumental in the generalization of CNNs to non-Euclidean domains. As we have already seen, any linear shift-invariant\(^1\) (Toeplitz) operator $W$ can be represented as the convolution $Wx = x \ast w$.

The action of any linear operator on a vector consists of scaling and rotating the vector. However, there are some privileged directions where no rotation occurs; such directions are called the eigenvectors of the operator. Specifically, for Toeplitz operators, given the input signal

$$\phi^\xi_n = e^{i2\pi \xi^T n},$$

parametrized by the vector $\xi \in [0,1]^d$, the output of the operator is

$$\left(W \ast \phi^\xi\right)_n = \sum_k w_n \phi_{n-k}^\xi = \sum_k w_k e^{i2\pi \xi^T (n-k)} e^{-i2\pi \xi^T k}.$$

Recalling the standard inner product on $\ell^2$,

$$\langle x, y \rangle = \sum_k x_k y_k^*,$$

we can express

$$\sum_k w_k e^{-i2\pi \xi^T k} = \sum_k w_k \phi_{-k}^\xi = \sum_k w_k \left(\phi_k^\xi\right)^* = \langle w, \phi^\xi \rangle = \hat{w}(\xi).$$

In these terms, the output is given by

$$w \ast \phi^\xi = \hat{w}(\xi) \phi^\xi,$$

which means that $\phi^\xi$ is an eigenvector of $W$ with the corresponding eigenvalues $\hat{w}(\xi)$. Note that while the eigenvalues depend on the specific operator (embodied in the sequence $w$ called the kernel of the operator), the eigenvectors are always the same: $\{\phi^\xi\}_{\xi \in [0,1]^d}$.

---

\(^1\)The term *shift-invariant* is so abundant in the signal processing and machine learning literature that we will not even attempt to change this unfortunate fact. However, it is worth noting that the correct mathematical term would be *shift-equivariant*. In general, let $f : \mathbb{U} \to \mathbb{V}$ be an operator mapping from some domain $\mathbb{U}$ to some co-domain $\mathbb{V}$, and let $\mathcal{G}$ be a group of transformations that can be applied both to the domain and the co-domain. The operator $f$ is said *invariant* to the action of $\mathcal{G}$ if $f \circ \tau = f$ for every $\tau \in \mathcal{G}$. On the other hand, the operator is *equivariant* if $f \circ \tau = \tau \circ f$. 

1.2 Fourier transform

The function
\[ \hat{w}(\xi) = \langle w, \phi^\xi \rangle_{\ell^2(\mathbb{Z}^d)} = \sum_k w_k e^{-i2\pi \xi^T k} \]
is called the \textit{(forward) Fourier transform} of the sequence \( w \). It is customary to define the operator \( \mathcal{F} : \ell^2(\mathbb{Z}^d) \rightarrow L^2([0,1]^d) \) mapping \( w \) to \( \hat{w} \), and refer to the argument \( \xi \) of the latter as to \textit{frequency}. The inverse map \( \mathcal{F}^{-1} : L^2([0,1]^d) \rightarrow \ell^2(\mathbb{Z}^d) \), called the \textit{inverse Fourier transform}, is given by
\[ w = \mathcal{F}^{-1} \hat{w} = \int_{[0,1]^d} \hat{w}(\xi) \phi^\xi d\xi = \int_{[0,1]^d} \hat{w}(\xi) e^{i2\pi \xi^T n} d\xi. \]

To prove this, observe that
\[ \int_{[0,1]^d} \hat{w}(\xi) e^{i2\pi \xi^T n} d\xi = \int_{[0,1]^d} \left( \sum_k w_k e^{-i2\pi \xi^T k} \right) e^{i2\pi \xi^T n} d\xi = \sum_k w_k \left( \int_{[0,1]^d} e^{i2\pi \xi^T (n-k)} d\xi \right). \]
Since the complex exponentials \( e^{i2\pi \xi^T (n-k)} \) have an integer number of periods on the domain \([0,1]^d\), the latter integral is zero unless \( n - k = 0 \), in which case it is exactly 1. Hence,
\[ \int_{[0,1]^d} \hat{w}(\xi) e^{i2\pi \xi^T n} d\xi = \sum_k w_k \delta_{n-k} = w_n. \]

Note that the inverse Fourier transform can also be written as
\[ (\mathcal{F}^{-1} \hat{w})_n = \langle \hat{w}, (\phi^n)^* \rangle_{L^2(\mathbb{Z}^d)}, \]
which emphasizes the above orthonormality property by essentially stating that \( \mathcal{F}^{-1} = \mathcal{F}^* \), where \( \mathcal{F}^* \) denotes the adjoint operator. Geometrically, this means that the Fourier transform is \textit{unitary}, a generalized form of rotation. The fact that rotations preserve distances leads to the celebrate \textit{Plancherel identity}
\[ \langle f, g \rangle_{\ell^2(\mathbb{Z}^d)} = \langle \hat{f}, \hat{g} \rangle_{L^2([0,1]^d)}, \]
and, in the particular case of \( g = f \), even more celebrate \textit{Parseval’s identity}
\[ \|f\|_{\ell^2(\mathbb{Z}^d)} = \|\hat{f}\|_{L^2([0,1]^d)}. \]

We can think of the Fourier transform as the transformation of a signal \( x \) to the (joint) eigenbasis of (all) Toeplitz operators, that is, describing \( x \) as a linear combination of the eigenvectors \( \phi^\xi \),
\[ x = \int_{-\pi}^{\pi} \hat{x}(\xi) \phi^\xi d\xi, \]
with $\hat{x}(\xi)$ serving as the coordinates in the eigenbasis. In the signal processing parlance, $x$ in the standard basis is called the impulse response, while in the Fourier basis it is referred to as the frequency response. The act of converting the impulse response to the frequency response is referred to as analysis, while the inverse is referred to as synthesis.

The notion of the Fourier transform allows to apply the operator $W$ in its eigenbasis. Indeed, describing the input sequence as a linear combination of the eigenvectors,

$$x = \int_{-\pi}^{\pi} \hat{x}(\xi) \phi^\xi d\xi,$$

we obtain

$$w \ast x = \mathcal{W} \int_{[0,1]^d} \hat{x}(\xi) \phi^\xi d\xi = \int_{[0,1]^d} \hat{x}(\xi) \mathcal{W} \phi^\xi d\xi = \int_{[0,1]^d} \hat{w}(\xi) \hat{x}(\xi) \phi^\xi d\xi.$$ 

The latter can be written as

$$w \ast x = \mathcal{F}^{-1}(\mathcal{F}w \cdot \mathcal{F}x)$$

or, schematically,

$$w \ast x \leftrightarrow \hat{w} \cdot \hat{x}.$$ 

This result is known as the convolution theorem, stating that convolution becomes pointwise product in the Fourier (frequency) domain. This result is the consequence of the fact that the Fourier transform diagonalizes Toeplitz operators (convolution). In fact, operating in the Fourier domain, we can define any Toeplitz operator as a diagonal linear operator (point-wise product), fully defined by the function $\hat{w}(\xi)$.

## 2 Non-Euclidean domains

In order to generalize the notion of a CNN to the non-Euclidean case, let us first define two types of non-Euclidean domains: manifolds and graphs (we will think of the latter as of some sort of discretization of the former).

### 2.1 Manifolds

A topological space $\mathcal{M}$ is called a $d$-dimensional manifold if every point $p$ in it has a neighborhood topologically equivalent (homeomorphic) to $\mathbb{R}^d$. The latter space is referred to as the tangent space at point $p$, denoted as $T_p\mathcal{M}$. The disjoint union of all tangent spaces is called the tangent bundle, denoted as $T\mathcal{M}$. Since each point $p$ is now associated with a linear space $T_p\mathcal{M}$, we can endow the latter with an inner product $\langle \cdot, \cdot \rangle_{T_p\mathcal{M}} : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ which we assume to depend smoothly on $p$, without further defining precisely what it means. This inner product is called a Riemannian metric and a manifold endowed with it is called a Riemannian manifold. The metric allows to (locally) measure lengths and angles.
Fields  A scalar field on $\mathcal{M}$ is a function of the form $f : \mathcal{M} \to \mathbb{R}$. A (tangent) vector field is a map $F : \mathcal{M} \to T\mathcal{M}$ assigning to every point $p \in \mathcal{M}$ a tangent vector $F(p) \in T_p\mathcal{M}$. Tangent vectors formalize the notion of infinitesimal displacements that we routinely use in calculus on Euclidean domains. Next, we define the Hilbert spaces of scalar and vector fields on $\mathcal{M}$ through the following standard inner products:

$$\langle f, g \rangle_{L^2(\mathcal{M})} = \int_{\mathcal{M}} f(p)g(p)dp;$$

$$\langle F, G \rangle_{L^2(T\mathcal{M})} = \int_{\mathcal{M}} \langle F(p), G(p) \rangle_{T_p\mathcal{M}}dp,$$

where the integration is performed w.r.t. the $d$-dimensional volume element $dp$ induced by the metric.

Differential  The notion of a derivative in calculus describes how the value of a function changes with an infinitesimal change of its argument. One of the big differences distinguishing calculus from differential geometry is a lack of a global vector space structure on the manifold, making expressions like $f(p + dp)$ meaningless. The conceptual leap that is required to generalize calculus to manifolds is the need to express all notions locally in the tangent spaces.

In order to construct calculus on a manifold, we define the differential of $f$ as the operator $df : T\mathcal{M} \to \mathbb{R}$ on tangent vectors. At every $p \in \mathcal{M}$, the differential is defined as the linear functional (a.k.a. 1-form in the differential geometry jargon)

$$df(p) : v \mapsto \langle \nabla f(p), v \rangle_{T_p\mathcal{M}},$$

$v \in T_p\mathcal{M}$. A vector field $F$ generalizes the notion of small displacements. In fact, we can write

$$df(p)F(p) = \langle \nabla f(p), F(p) \rangle_{T_p\mathcal{M}},$$

as the extension of the regular notion of directional derivative in Euclidean spaces,

$$df = f(p + dp) = \langle \nabla f(p), dp \rangle = \frac{\partial f(p)}{\partial p_1}dp_1 + \cdots + \frac{\partial f(p)}{\partial p_d}dp_d.$$

Gradient and divergence  The operator $\nabla f(p) : \mathcal{M} \to T\mathcal{M}$ appearing in the definition of the differential generalizes the notion of the gradient defining the direction of the steepest increase of $f$; the main difference is that on a manifold the latter direction is given by tangent vector. The gradient can be viewed as an operator of the form $\nabla : L^2(\mathcal{M}) \to L^2(T\mathcal{M})$ mapping scalar fields to vector fields. Its adjoint is called the divergence operator, $\text{div} f(p) : L^2(T\mathcal{M}) \to L^2(\mathcal{M})$ mapping vector fields to scalar fields and satisfying

$$\langle F, \nabla f \rangle_{L^2(T\mathcal{M})} = \langle \nabla^* F, f \rangle_{L^2(\mathcal{M})} = \langle -\text{div} F, f \rangle_{L^2(\mathcal{M})}$$

(note the minus sign!). As vector fields can be thought of as a model of a flow on the manifold, the divergence operator measures the net flow at a point.
Laplacian  The *Laplacian* (a.k.a. the *Laplace-Beltrami operator*) $\Delta : L^2(M) \to L^2(M)$ is defined as

$$\Delta = \nabla^* \nabla = -\text{div} \nabla.$$

The Laplacian of a scalar field $f$ at point $p$ can be interpreted as the difference between the average value of the field on an infinitesimal sphere around $p$ and the value of $f(p)$.

By virtue of the adjoint relation between the gradient and the negative divergence, the Laplacian is self-adjoint (symmetric), that is, for every scalar field $f$,

$$\langle \nabla f, \nabla f \rangle_{L^2(TM)} = \langle \nabla^* \nabla f, f \rangle_{L^2(M)} = \langle \Delta f, f \rangle_{L^2(M)}$$

and

$$\langle \nabla f, \nabla f \rangle_{L^2(TM)} = \langle f, \nabla^* \nabla f \rangle_{L^2(M)} = \langle f, \Delta f \rangle_{L^2(M)}.$$

The expression $\langle \Delta f, f \rangle_{L^2(M)}$ is known as the *Dirichlet energy* of the field $f$ and measures the “smoothness” of the field on $M$. Physically, it can be interpreted as the potential energy due to the bending of an elastic body.

2.2 Graphs

We will limit our attention to undirected graphs and view them as a discrete analog of manifolds. We define the *vertex set* $V = \{1, \ldots, n\}$ (it can be any set containing $n$ objects, which we canonically map to the above set of natural numbers from 1 to $n$); the *edge set* and the *edge set* $E \subseteq V \times V$. An undirected graph has $(i, j) \in E \iff (j, i) \in E$. We further define the vertex weights as the function $a : V \to (0, \infty)$ and the edge weights as $w : E \to \mathbb{R}_+$ (in fact, $w$ can be defined on the entire $V \times V$ with $w_{ij} = 0$ meaning $(i, j) \notin E$). We refer to the tuple $G = (V, E, a, w)$ as to a *weighted undirected graph*.

**Difference operators**  A *vertex field* is a function of the form $f : V \to \mathbb{R}$, while an *edge field* is a function of the form $F : E \to \mathbb{R}$. Vertex and edge fields on a graph are the discrete analogs of scalar and vector fields on a manifold (under the tacit assumption that $F_{ij} = -F_{ji}$ for technical reason we are not going to detail). As in the case of manifolds, we define the two Hilbert spaces, $\ell^2(V)$ and $\ell^2(E)$ through the corresponding inner products

$$\langle f, g \rangle_{\ell^2(V)} = \sum_{i \in V} a_i f_i g_i;$$

$$\langle F, G \rangle_{\ell^2(E)} = \sum_{(i,j) \in E} w_{ij} F_{ij} G_{ij};$$

note that the weights play the role of discrete volume elements we had before in the integrals on manifolds.

The graph *gradient* is the operator $\nabla : \ell^2(V) \to \ell^2(E)$ defined by

$$(\nabla f)_{ij} = f_i - f_j.$$
Note that the resulting edge field is, by definition, alternating, that is, \((\nabla f)_{ij} = -(\nabla f)_{ji}\). Analogously to manifolds, the adjoint operator, the graph divergence \(\text{div} : \ell^2(E) \rightarrow \ell^2(V)\) is defined as
\[
(\text{div } F)_i = \frac{1}{a_i} \sum_{(i,j) \in E} w_{ij} F_{ij}.
\]
It is straightforward to verify that
\[
\langle F, \nabla f \rangle_{\ell^2(E)} = \langle \nabla^* F, f \rangle_{\ell^2(V)} = \langle -\text{div } F, f \rangle_{\ell^2(V)}.
\]

**Graph Laplacian** Having the gradient and the divergence operators defined, we define the graph Laplacian \(\Delta : \ell^2(V) \rightarrow \ell^2(V)\) as \(\Delta = \nabla^* \nabla = -\text{div } \nabla\), or, explicitly,
\[
(\Delta f)_i = \frac{1}{a_i} \sum_{(i,j) \in E} w_{ij} (f_i - f_j).
\]
Observe how this expression manifests the meaning of the Laplacian as the difference between the value of a field at a vertex and the (weighed) average of its values in the surrounding.

Since the vertex set is finite, it is convenient to represent the Laplacian as an \(n \times n\) matrix. For that purpose, we denote the edge weights by the \(n \times n\) matrix \(W = (w_{ij})\), the vertex weights by the diagonal matrix \(A = \text{diag}\{a_1, \ldots, a_n\}\), and by \(D = \text{diag}\left\{\sum_{j:j \neq i} w_{ij}\right\}\) the vertex degree matrix. In this notation, the graph Laplacian is given by
\[
\Delta = A^{-1}(D - W).
\]
Different choices of \(A\) lead to different definitions of a Laplacian. For \(A = I\), the unnormalized graph Laplacian
\[
\Delta_{un} = D - W
\]
is obtained. The choice \(A = D^{-1}\) leads to the random walk Laplacian
\[
\Delta_{rw} = I - D^{-1} W.
\]
The term \(D^{-1} W\) in the definition of the above operator can be interpreted as a transition probability of random walks on the graph, hence the name. Finally, when the graph is used as a discrete approximation of the underlying continuous manifold (as is the case of simplicial complexes a.k.a. meshes), its weight matrices \(A\) and \(W\) are obtained from the discretized metric of the manifold.

### 2.3 Fourier transform on non-Euclidean domains

Thus far, we have constructed two types of non-Euclidean domains, manifolds and graphs, that both had a similarly defined Laplacian operator. Next, we are going to use the Laplacian
to define an analog of Fourier analysis. For convenience, we are going to construct the Fourier transform on manifolds; the construction for graphs is straightforwardly similar.

The Laplacian, being a self-adjoint operator, admits an **orthogonal eigendecomposition**

\[ \Delta \phi_i = \lambda_i \phi_i. \]

The eigenvalues \( \lambda_i \) (called the **spectrum** of the Laplacian) can be furthermore shown to be non-negative, a manifestation of the fact that the Laplacian is a positive semi-definite operator (by analogy, think of a matrix defined through \( \Delta = \nabla^T \nabla \)). On Euclidean domains, the eigenfunctions of the Laplacian are simply complex exponentials.

A scalar field \( f \in L^2(\mathcal{M}) \) can be represented in the Laplacian eigenbasis as

\[ f = \sum_{i \geq 0} \hat{f}_i \phi_i \]

with the coordinates \( \hat{f} = \{ \hat{f}_i \}_{i \in \mathbb{Z}} \). Because of orthonormality of the eigenfunctions, the coefficients \( \hat{f}_i \) are given by

\[ \hat{f}_i = \langle f, \phi_i \rangle_{L^2(\mathcal{M})}. \]

We will call the operator \( \mathcal{F} : L^2(\mathcal{M}) \to \ell^2(\mathbb{Z}) \) defined as

\[ \mathcal{F}f = \{ \langle f, \phi_i \rangle_{L^2(\mathcal{M})} \}_{i \in \mathbb{Z}} \]

as the **Fourier transform** (analysis) on \( \mathcal{M} \). The inverse (synthesis) transform \( \mathcal{F}^{-1} : \ell^2(\mathbb{Z}) \to L^2(\mathcal{M}) \) is given by

\[ \mathcal{F}^{-1} \hat{f} = \sum_{i \geq 0} \hat{f}_i \phi_i. \]

As before, it is easy to show that the above two operations are adjoint w.r.t. the standard inner products on \( L^2(\mathcal{M}) \) and \( \ell^2(\mathbb{Z}) \).

### 2.4 Convolution on non-Euclidean domains

Recall that one of the principal properties the Fourier transform enjoyed on Euclidean domains was the fact that it diagonalized Toeplitz operators. In fact, we had the property

\[ \mathcal{F}(f * g) = \mathcal{F}f \cdot \mathcal{F}g. \]

Unfortunately, the absence of a trivially defined translation group on general non-Euclidean domains does not allow to generalize convolution, which makes the left-hand-side of the above equation undefined. However, the right-hand-side, being simply an element-wise product of frequency responses, is perfectly defined, so we will use it to define convolution on the non-Euclidean domain as

\[ f * g = \mathcal{F}^{-1}(\mathcal{F}f \cdot \mathcal{F}g) = \sum_{i \geq 0} \langle f, \phi_i \rangle_{L^2(\mathcal{M})} \langle g, \phi_i \rangle_{L^2(\mathcal{M})} \phi_i. \]
The standard shift-invariance (or, more precisely, translation-equivariance) property of convolution on Euclidean domains is lost of course. Using a signal processing metaphor, it can be interpreted as a position-dependent filter, with the impulse response that can differ significantly at different locations in the domain.

When dealing with discrete domain such as graphs (which we will henceforth assume for convenience), the Fourier transform and its inverse have a matrix form. Note that the eigendecomposition of the Laplacian $\Delta$ can be written as $\Delta = \Phi \Lambda \Phi^T$, where $\Phi$ has the eigenvectors as its columns and $\Lambda = \text{diag}\{\lambda_1, \ldots, \lambda_n\}$. Representing vertex fields as $n$-dimensional column vectors, the analysis (the forward transform) can be written as

$$\hat{f} = \mathcal{F}f = \Phi^T f = (\langle f, \phi_1 \rangle, \ldots, \langle f, \phi_n \rangle)^T;$$

likewise, the synthesis operator (the inverse transform) assume the form

$$f = \mathcal{F}^{-1} \hat{f} = \Phi \hat{f} = \phi_1 \hat{f}_1 + \cdots + \phi_n \hat{f}_n.$$

In this notation, the convolution of two fields $f$ and $g$ can be written as

$$f \ast g = \Phi ((\Phi f) \odot (\Phi g)),$$

where $\odot$ denotes the Hadamard (element-wise) product.

Convolution on Euclidean domains was an operation commuting with any translation-equivariant (Toeplitz) operator, including the Laplacian. In generalizing it to non-Euclidean domains, we only demanded commutativity with the Laplacian.

## 3 Spectral CNN

The spectral definition of a convolution-like operation on a non-Euclidean domain allows to parametrize the action of a filter as

$$Wf = \Phi \hat{W} \Phi^T f,$$

where $\hat{W}$ is a diagonal weight matrix containing the filter’s frequency response on the diagonal. In the space domain, it amounts to applying the operator $W = \Phi \hat{W} \Phi^T$ to $f$, by computing the inner products of $f$ with every row of $W$ and stacking the resulting numbers into a vertex field. Different weight matrices $\hat{W}$ realize different such operators.

Note that the definition is basis-dependent: a change in the domain, and, consequently, in $\Phi$ may translate the same $\hat{W}$ into a completely different operator. Therefore, this construction must assume the domain fixed; if we learn the weights $\hat{W}$, they will typically generalize rather poorly even to similarly-looking domains. Such a complication did not exist on Euclidean domains.

Armed with the notion of a generalized convolution on non-Euclidean domains, we can mimick the construction of a regular CNN. For this purpose, we construct a spectral convolutional layer accepting an $m$-dimensional vertex field $x = (x^1, \ldots, x^m)$ and outputting an
\( m' \)-dimensional vertex field \( y = (y^1, \ldots, y^{m'}) \), whose \( i \)-th dimension is defined according to

\[
y_j = \varphi \left( \sum_{i=1}^{m} \Phi \hat{W}^{ij} \Phi^T x^i \right),
\]

where \( \varphi \) is an element-wise non-linearity such as ReLU, and \( \hat{W}^{ij} \) are diagonal matrices parametrizing the filters of the layer.

### 3.1 Strided convolution

Recall that a typical Euclidean CNN architecture used strided convolutions of the form

\[
(\downarrow_p (w * x))_k = (w * x)_{p \odot k} = \sum_{(i_1, \ldots, i_d)} w_{i_1 \ldots i_d} x_{p_1 k_1 - i_1 \ldots p_d k_d - i_d},
\]

where \( p = (p_1, \ldots, p_d) \) is a \( d \)-dimensional vector of strides. This can be thought of projecting the result of the convolution \( w * x \) performed on \( \mathbb{Z}^d \) onto the coarser domain \( \downarrow_p \mathbb{Z}^d \). The subsampling operator \( \downarrow_p \) can be thought of as a projection of a signal on \( \mathbb{Z}^d \) onto \( \downarrow_p \mathbb{Z}^d \).

The non-Euclidean analog can be constructed along the same lines. Let \( \mathcal{G} \) be the original domain of size \( n \) with the Laplacian \( \Delta = \Phi \Lambda \Phi^T \), and let \( \tilde{\mathcal{G}} \) be its coarsened (sub-sampled) version containing \( \tilde{n} = \alpha n < n \) vertices. We denote by \( \tilde{\Delta} = \tilde{\Phi} \tilde{\Lambda} \tilde{\Phi}^T \) the corresponding Laplacian and its eigendecomposition. To keep the previous notation, we denote by \( \downarrow_\alpha \): \( \mathcal{G} \rightarrow \tilde{\mathcal{G}} \) the projection onto the coarse domain, i.e., \( \downarrow_\alpha \) maps a vertex field on \( \mathcal{G} \) to a vertex field on \( \tilde{\mathcal{G}} \). In matrix form, \( \downarrow_\alpha \) is an \( \tilde{n} \times n \) matrix whose \( i \)-th row encodes the position of the \( i \)-th vertex of the coarse domain \( \tilde{\mathcal{G}} \) in the fine domain \( \mathcal{G} \).

The eigenvectors \( \Phi \) and \( \tilde{\Phi} \) of the fine and the coarse Laplacians, \( \Delta \) and \( \tilde{\Delta} \), satisfy the following multi-resolution property:

\[
\tilde{\Phi} \approx \downarrow_\alpha \Phi P_\alpha,
\]

where the \( n \times \tilde{n} \) matrix

\[
P_\alpha = \begin{pmatrix} I_{\alpha n} \\ 0 \end{pmatrix}
\]

denotes the projection onto the lowest \( \tilde{n} = \alpha n \) frequencies. This property essentially means that only the first \( k = \alpha n \) components of the spectrum can be retained. Thus, the strided convolutional layer assumes the form

\[
y_j = \varphi \left( \sum_{i=1}^{m} \tilde{\Phi}_k \hat{W}^{ij} \tilde{\Phi}_k^T x^i \right),
\]

where \( \tilde{\Phi}_k = (\phi_1, \ldots, \phi_k) \) is the truncated eigenbasis of the fine Laplacian containing the first \( k \) eigenvectors, and the weight matrices \( \hat{W}^{ij} \) are now \( \tilde{n} \times \tilde{n} \). The layer accepts an \( m \)-dimensional vertex field \( x = (x^1, \ldots, x^m) \) on \( \mathcal{G} \) as the input and produces an \( m' \)-dimensional vertex field \( y = (y^1, \ldots, y^{m'}) \) on \( \tilde{\mathcal{G}} \) as the output.
3.2 Spatial localization

Note that in our construction of a spectral convolutional layer, each weight matrix has \( k = \mathcal{O}(n) \) degrees of freedom, so that each layer has \( \mathcal{O}(nmm') \) degrees of freedom, unlike the regular CNN, in which the layer was parametrized in the spatial domain by a fixed-size kernel with the number of parameters independent on the domain size \( n \). In order to keep the number of parameters under control and avoid overfitting, we would like to impose spatial localization onto the weights \( \hat{W}^{ij} \), that is, ensure that the vertex fields defined by every row of the operator \( \hat{W} = \Phi W \Phi^T \) are spatially localized.

On a Euclidean domain, the spatial localization of a signal \( w : \mathbb{Z} \to \mathbb{R} \) is controlled by the decay of its moments, defined as

\[
\mu_p^2(w) = \sum_{k \in \mathbb{Z}} k^{2p} w_k^2 = \| k^p . w_k \|_{\ell^2(\mathbb{Z})}^2.
\]

The faster \( \mu_p^2(w) \) vanishes as \( p \) increases, the more localized is \( w \). From

\[
\frac{\partial}{\partial \xi} \hat{F}w = \frac{\partial}{\partial \xi} \left( \sum_{k \in \mathbb{Z}} w_k e^{-i2\pi \xi k} \right) = \sum_{k \in \mathbb{Z}} -i2\pi k w_k e^{-i2\pi \xi k} = -i2\pi F(kw_k)
\]

we obtain the property

\[
k^p . w_k \xrightarrow{\hat{F}} \left( \frac{i}{2\pi} \right)^p \frac{\partial^p \hat{w}}{\partial \xi^p}.
\]

Invoking Parseval’s identity,

\[
\mu_p^2(w) = \| k^p . w_k \|_{\ell^2(\mathbb{Z})}^2 = \left\| \left( \frac{i}{2\pi} \right)^p \frac{\partial^p \hat{w}}{\partial \xi^p} \right\|_{L^2([0,1])}^2 = \frac{1}{(2\pi)^{2p}} \int_{[0,1]} \left| \frac{\partial^p \hat{w}(\xi)}{\partial \xi^p} \right|^2 d\xi.
\]

This result implies that fast decay of \( \mu_p^2(w) \) implies fast decay of the derivatives of \( \hat{w} \), or, said differently, localization in the spatial domain is equivalent to smoothness in the frequency domain (the fact that smoothness is opposite to localization brings forth the renowned Heisenberg’s uncertainty principle). Smoothness of the frequency response \( \hat{w} \) can be asserted by representing it in an underdetermined smooth basis or, equivalently, specifying it only at a small set of frequencies and completing the rest via some smooth interpolation.

This idea can be generalized to non-Euclidean domains. The only complication is that while in \( \mathbb{R}^d \) we had a trivial notion of smoothness arising in the spectrum, since the similarity between two basis functions \( \phi^\xi = e^{i2\pi x^T \xi} \) and \( \phi'^\xi = e^{i2\pi x^T \xi'} \) could be quantified as the distance \( \| \xi - \xi' \| \), there is not such a standard notion in the spectrum of a general non-Euclidean domain. A formal way to define smoothness is by constructing a dual graph whose weights \( w^*_{ij} \) reflect the similarity between the eigenvectors \( \phi_i \) and \( \phi_j \) of the Laplacian of the original (primal) graph. The question of how to define such a dual graph the smoothness on which will lead to maximal localization on the primal graph is still open. However, empirical
evidence shows that at least in some cases, the simple definition of \( w_{ij}^* = |\lambda_i - \lambda_j| \) leads to reasonable localization.

With this notion of smoothness in mind, we fix a set of \( q \) smooth basis functions \( \beta_1(\lambda), \ldots, \beta_q(\lambda) \) (e.g., cubic splines) and sample them at \( \lambda \in \{\lambda_1, \ldots, \lambda_k\} \). We arrange the samples into a \( k \times q \) matrix \( B \) with the elements \( b_{rs} = \beta_s(\lambda_r) \). The spectral weight matrices \( \hat{W}_{ij} \) can now be defined as

\[
\hat{W}_{ij} = \text{diag}\{B\alpha_{ij}\},
\]

where \( \alpha_{ij} \) are \( q \)-dimensional interpolation coefficients. In order to render the layer complexity independent of the domain size, one has to choose \( q = \mathcal{O}(1) \).

### 4 Spatial CNN

One of the main disadvantages of the spectral construction of a convolutional layer is its high computational complexity. The multiplication by \( \Phi \) and \( \Phi^T \) in the forward and backward passes require \( \mathcal{O}(n^2) \) operations, which quickly becomes prohibitively expensive for large domains. Unlike Euclidean domains on which the forward and inverse Fourier transforms can be carried out using FFT in \( \mathcal{O}(n \log n) \) operations, no such fast algorithms exist for general non-Euclidean domains. In what follows, we will reformulate the convolutional layer in a way free of the costly Laplacian eigendecomposition and explicit projection on its basis.

Let us substitute \( \hat{W} = \text{diag}\{B\alpha\} \) and examine the spatial representation of the linear part of the layer:

\[
W = \Phi_k \hat{W} \Phi_k^T = \Phi_k \begin{pmatrix}
\sum_{i=1}^{q} \alpha_i \beta_i(\lambda_1) & \cdots & \\
& \ddots & \\
& & \sum_{i=1}^{q} \alpha_i \beta_i(\lambda_n)
\end{pmatrix} \Phi_k^T
\]

(note that we assumed \( \beta_i(\lambda) = 0 \) for \( \lambda > \lambda_k \)). Denoting by

\[
b(\lambda) = \sum_{i=1}^{q} \alpha_i \beta_i(\lambda),
\]

we have

\[
W = \Phi \text{diag}\{b(\lambda_1), \ldots, b(\lambda_n)\} \Phi^T.
\]

(note that we assumed \( b(\lambda) = 0 \) for \( \lambda > \lambda_k \)).

Since \( b(\lambda) \) is typically a polynomial (of degree 3 in case of cubic splines), let us examine how to rewrite it directly in the spatial domain. Let \( \Delta = \Phi \Lambda \Phi^T \) be the eigendecomposition of the Laplacian, and suppose we would like to compute \( \Delta^p \) for some integer power \( p \). Then,

\[
\Delta^p = \Phi \Lambda \Phi^T \cdots \Phi \Lambda \Phi^T = \Phi \Lambda^p \Phi^T = \Phi \text{diag}\{\lambda_1^p, \ldots, \lambda_n^p\} \Phi^T.
\]
Using linearity, we can conclude that for any polynomial

\[ b(\lambda) = \sum_{i=0}^{r} \alpha_i \lambda^i, \]

one has

\[ b(\Delta) = \sum_{i=0}^{r} \alpha_i \Delta^i = \Phi \text{ diag } \{b(\lambda_1), \ldots, b(\lambda_n)\} \Phi^T. \]

In other words, the expensive right-hand-side can be simply evaluated as applying the polynomial \( b \) directly to the Laplacian. The Laplacian is typically a sparse \( n \times n \) matrix with \( O(1) \) non-zero entries in every row. In such cases, computing its powers takes \( O(n) \) operations, and the entire calculation is \( O(nr) \). Also note that since the Laplacian is a local operator acting on 1-rings, its highest power \( \Delta^r \) will act on \( r \)-rings, keeping the operator \( b(\Delta) \) spatially localized.

Using this observation, we can reformulate the convolutional layer directly in the spatial domain as

\[ y_j = \varphi \left( \sum_{i=1}^{m} \sum_{k=0}^{r} \alpha_k^{ij} \Delta^k x^i \right), \]