Graph and Simplicial Complex Prediction Gaussian Process via Hodgelet Representations

Mathieu Alain[1†] **So Takao**[2] **Bastian Rieck**[3] **Xiaowen Dong**[4] **Emmanuel Noutahi**[5]
[1]University College London [2]PhysicsX [3]University of Fribourg
[4]University of Oxford [5]Valence Labs
[†]mathieu.alain.21@ucl.ac.uk

Abstract

Predicting labels for graph-structured data is crucial in many scientific applications. Recently, Gaussian processes with graph-level inputs have been proposed as flexible non-parametric models for classification tasks. In this work, we extend this framework to regression tasks and to simplicial complexes, enabling the modelling of attributes defined on edges and higher-order simplices. Drawing on Hodge theory, we construct representations based on the Hodge decomposition, allowing the model to distinguish between exact, co-exact, and harmonic signal components. We introduce Hodgelet representations as rich, learnable topological descriptors of simplicial complexes and show that our framework improves predictive performance across a range of applications. Our work broadens the applicability of Gaussian processes to prediction tasks at the graph and simplicial-complex levels.

1 Introduction and motivation

In recent years, considerable effort has been devoted to adapting machine learning methods to non-Euclidean domains such as graphs [Bronstein et al., 2021, Papamarkou et al., 2024]. This setting presents numerous challenges due to the irregular structure of graphs, their varying sizes, and the information supported on different graph elements, such as vertices and edges. Predicting graph-level labels, whether continuous or discrete, arises in applications ranging from estimating molecular properties [Reiser et al., 2022] to classifying textual documents [Rousseau et al., 2015].

Simplicial complexes (SCs) generalise graphs by incorporating higher-order interactions. Whereas graphs encode pairwise interactions through edges, SCs contain simplices that can represent higher-order interactions among vertices. Of particular interest are simplicial 2-complexes, which extend graphs by including 2-simplices, i.e., triangular faces (filled triangles) formed by triples of vertices. Among their many applications, simplicial complexes can represent two-dimensional meshes [Ballester et al., 2025] and higher-order interactions in protein–protein interaction networks [Malod-Dognin and Pržulj, 2019]. Moreover, any graph can be extended to a simplicial 2-complex, for example, by filling its 3-cliques with 2-simplices, enabling a richer topological representation [Bodnar et al., 2021].

Graph neural networks (GNNs) are widely used for prediction tasks on graphs. However, they can require substantial amounts of training data, may be difficult to interpret despite ongoing efforts to develop explainability methods [Ying et al., 2019, Luo et al., 2020], and can be prone to overfitting, particularly in data-scarce settings [Rong et al., 2020]. Increasing GNN depth can also lead to over-smoothing [Li et al., 2018]. Furthermore, capturing global graph properties through message passing may require many layers, especially for graphs with large diameters or structural bottlenecks.

Graph kernels provide an alternative [Borgwardt et al., 2020]. Despite the prevalence of GNNs, graph-kernel methods can achieve competitive performance [Kriege et al., 2020]. Although these

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

methods can be more data-efficient and interpretable, designing effective graph kernels remains challenging. Many existing graph kernels focus on local structural patterns, can be computationally expensive for large graphs, struggle to accommodate multidimensional vertex attributes, or rely on hand-crafted features to construct representations of graphs with varying sizes.

Another option is to construct a fixed-dimensional representation of each graph and apply a standard Euclidean kernel, such as the radial basis function (RBF) kernel, also known as the squared-exponential kernel. However, this shifts the challenge to constructing a suitable representation that captures both the graph topology and its associated attributes. Such representations can then be naturally combined with Gaussian processes (GPs) [Rasmussen and Williams, 2005] for graph-level prediction. GPs are flexible non-parametric models that provide principled predictive uncertainty and can perform well in small-to-medium data regimes. This makes them particularly attractive when training data are limited by restricted access or the cost of data acquisition. Furthermore, the parameters used to construct the graph representations can be optimised jointly with the kernel hyperparameters by maximising the marginal likelihood.

A promising approach to capturing information from graph vertex attributes while accounting for the underlying topology is to leverage techniques from graph signal processing [Ortega et al., 2018]. In particular, the graph wavelet transform [Hammond et al., 2011] provides a framework for analysing graph signals with localisation in both the spatial and spectral domains. In Opolka et al. [2023], wavelet coefficients obtained from vertex attributes are aggregated into fixed-dimensional graph representations, which are then used as inputs to a GP with a standard Euclidean kernel. The wavelet parameters are optimised jointly with the kernel hyperparameters, allowing the model to emphasise task-relevant graph frequencies. Although this approach effectively incorporates vertex attributes, it does not account for attributes defined on edges, which may encode important information such as flow magnitudes, interaction strengths, or binding affinities.

This limitation motivates the use of methods from topological signal processing [Barbarossa and Sardellitti, 2020], such as the simplicial wavelet transform [Roddenberry et al., 2022], to construct representations incorporating both vertex and edge attributes. By exploiting the Hodge decomposition, simplicial wavelet transforms can be applied separately to three fundamental signal components: exact, co-exact, and harmonic. These transforms yield Hodgelet coefficients that capture gradient-like, curl-like, and harmonic contributions, respectively. When the signals are simplicial cochains, these coefficients are equivariant under simplex relabellings and changes of orientation, but not invariant. By aggregating the coefficients using permutation-invariant functions and concatenating the resulting summaries across multiple filters, we construct fixed-dimensional simplicial representations, which we call Hodgelet representations. For cochains, an aggregation that is also invariant under sign changes makes these representations invariant.

Previous studies have shown that processing Hodge components separately can improve predictive performance [Battiloro et al., 2023, Yang et al., 2024, Isufi et al., 2025]. The Hodge decomposition distinguishes signal components associated with different aspects of the underlying simplicial structure. In particular, the harmonic subspace reflects global topological properties, with its dimension corresponding to the number of independent k-dimensional holes in the simplicial complex. Modelling these components separately allows the model to exploit their distinct characteristics.

Our work extends graph-level GP representations by incorporating edge attributes and exploiting the Hodge decomposition. Across synthetic vector-field, molecular, and power-grid benchmarks, we demonstrate that our approach achieves predictive performance competitive with or better than that of GNN baselines in small-to-medium data regimes, often with better calibration, without requiring extensive feature or architectural engineering. We focus primarily on simplicial 2-complexes, consisting of vertices, edges, and triangular faces, which suffice for the applications considered here. Nevertheless, the framework extends to simplicial complexes of arbitrary finite dimension and can be adapted to other structures equipped with suitable boundary operators, such as cellular complexes [Alain et al., 2024a]. Although particularly well suited to GPs, for which kernel and wavelet parameters can be optimised jointly by maximising the marginal likelihood or a variational objective, Hodgelet representations can also be used with other machine learning models. This work is the extension of the workshop paper [Alain et al., 2024b].

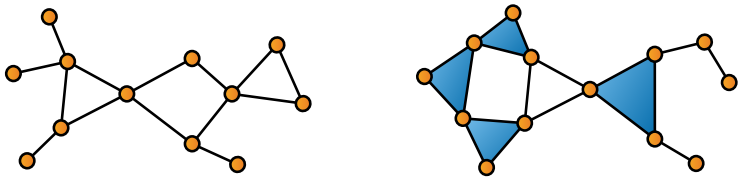
2 Simplicial complex prediction with Gaussian processes

In this section, we briefly introduce the concept of a simplicial complex and establish the framework for predicting its label using Gaussian processes.

2.1 Simplicial complexes

A k-*simplex* is a basic building block of a *simplicial complex* (SC). Simplices generalise vertices, edges, and triangles to arbitrary dimensions: a 0-simplex is a vertex, a 1-simplex is an edge, a 2-simplex is a filled triangle, and so on. SCs can therefore encode higher-order relations involving more than two vertices.

Let $K \geq 0$ denote the maximum simplex dimension considered, and let S_k denote a set of $N_k \geq 0$ simplices of dimension k, for $0 \leq k \leq K$. Of particular interest are the simplex sets S_0, S_1, and S_2, corresponding to vertices, edges, and triangular faces, respectively. We assume that the sets $S_0, \ldots, S_K$ form a simplicial complex of dimension at most K (see Figure 1(b)). Such sets must satisfy compatibility conditions to define a simplicial complex; we refer the reader to Appendix A and Barbarossa and Sardellitti [2020] for further details. A graph (see Figure 1(a)) can be viewed as a simplicial 1-complex.



(a) Simplicial 1-complex (graph). (b) Simplicial 2-complex.

Figure 1: Orange vertices are 0-simplices, black edges are 1-simplices, and blue triangles are 2-simplices.

Furthermore, for each simplex dimension k, we assume that every k-simplex is associated with a real-valued vector of dimension $D_k \geq 1$, specified by an attribute function defined on the simplex set S_k and taking values in $\mathbb{R}^{D_k}$. These attributes can be interpreted as *signals* on the simplices, a perspective that will be particularly useful when introducing the wavelet transform in Section 4.1.

Next, we fix an *orientation* for each simplex. The choice of orientation is arbitrary and can, for example, be induced by a total ordering of the vertices. This allows us to encode the incidence structure of the simplicial complex, that is, how $(k-1)$-simplices are incident to k-simplices, using the signed *incidence matrices* (also called *boundary matrices*) $\boldsymbol{B}_k \in \mathbb{Z}^{N_{k-1} \times N_k}$. For notational convenience, we set $\boldsymbol{B}_0 \coloneqq \boldsymbol{0}$ and $\boldsymbol{B}_k \coloneqq \boldsymbol{0}$ for $k > K$, with these zero matrices understood to have dimensions compatible with the expressions in which they appear. After assigning an index to each simplex, the attribute functions can be represented as the matrices $\boldsymbol{X}_k \in \mathbb{R}^{N_k \times D_k}$.

We treat each attribute dimension $1 \leq d \leq D_k$ separately, as different dimensions may encode quantities of different types. For example, molecular attributes may include atomic charge, atom type, bond length, or bond type. We therefore write $\boldsymbol{X}_k = [\boldsymbol{x}_{k1}, \ldots, \boldsymbol{x}_{kD_k}] \in \mathbb{R}^{N_k \times D_k}$, where the vector $\boldsymbol{x}_{kd} \in \mathbb{R}^{N_k}$ contains the values of the d-th attribute across all k-simplices. For ease of exposition, we henceforth restrict attention to $D_k = 1$, that is, to a scalar attribute on each k-simplex, and denote $\boldsymbol{X}_k$ simply by the attribute vector $\boldsymbol{x}_k \in \mathbb{R}^{N_k}$. We refer to $\mathbb{R}^{N_k}$ as the *attribute space*, and represent an *attributed simplicial complex* by the collection $\mathcal{K} \coloneqq \{(S_k, \boldsymbol{x}_k)\}_{k \leq K}$ of simplex sets and their associated attribute vectors.

Attributes defined on oriented simplices can be viewed as *simplicial cochains* when their values change sign under orientation reversal. For example, reversing the orientation of an edge changes the sign of the line integral of a vector field along that edge, whereas orientation-independent attributes, such as molecular bond lengths or bond types, remain unchanged. This distinction is important for the equivariance properties of the Hodgelet coefficients introduced in Section 4.1.

2.2 Gaussian process prediction of attributed SCs

Let $\mathcal{S}$ denote a set of attributed simplicial K-complexes and $\mathcal{Y}$ a set of labels. Given a dataset $\mathcal{D} \coloneqq \{(\mathcal{K}_i, y_i)\}_{i \leq m} \subseteq \mathcal{S} \times \mathcal{Y}$ consisting of m attributed simplicial complexes and their associated labels, our goal is to learn a predictive function $g : \mathcal{S} \to \mathcal{Y}$. We use Gaussian processes (GPs) to model this prediction problem [Rasmussen and Williams, 2005]. GPs provide a principled Bayesian framework for learning functions by placing a probability distribution over possible functions.

We introduce a latent function $f : \mathcal{S} \to \mathbb{R}$ endowed with a GP prior,

$$f \sim \mathcal{GP}(\mu, \kappa), \tag{1}$$

where $\mu : \mathcal{S} \to \mathbb{R}$ is the mean function and $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is the covariance kernel. We denote the prior distribution by $p(f)$. Both the mean function and the covariance kernel may depend on tunable hyperparameters, which often have interpretable roles, such as controlling the smoothness and variability of the latent function, in contrast to the typically less interpretable parameters of neural network models.

We further specify a likelihood $p(y_i \mid f(\mathcal{K}_i))$ for each $(\mathcal{K}_i, y_i) \in \mathcal{D}$. By Bayes' rule, the posterior distribution over f is

$$p(f \mid \mathcal{D}) \propto p(f) \prod_{i=1}^{m} p(y_i \mid f(\mathcal{K}_i)). \tag{2}$$

For a test simplicial complex $\mathcal{K}_* \in \mathcal{S}$, let $f_* \coloneqq f(\mathcal{K}_*)$. The posterior predictive distribution is obtained by marginalising over the posterior distribution of f_*:

$$p(y_* \mid \mathcal{K}_*, \mathcal{D}) = \int p(y_* \mid f_*)\, p(f_* \mid \mathcal{K}_*, \mathcal{D})\, df_*. \tag{3}$$

For regression tasks with Gaussian observation noise, both the posterior and posterior predictive distributions admit closed-form expressions. For classification tasks, the likelihood is non-Gaussian, and the posterior is generally intractable, requiring approximate inference (see Appendix B). While we present the scalar-valued formulation for simplicity, multiclass classification can be accommodated by introducing multiple latent functions. The hyperparameters of the GP, together with the parameters used to construct the simplicial complex representations, can be learned jointly by maximising the marginal likelihood or an appropriate variational objective.

This describes the general GP framework; it remains to specify how the topology and attributes of a simplicial complex are incorporated into the prediction model. In the following sections, we construct a representation of each simplicial complex by applying spectral wavelet transforms to its attributes. These representations then serve as inputs to the GP.

3 Hodge Laplacian and its decomposition

In this section, we introduce the discrete Hodge Laplacian and its associated decomposition. Recall from Section 2.1 that, once oriented, a simplicial complex is equipped with incidence matrices $\boldsymbol{B}_k$, representing discrete boundary operators. Their transposes $\boldsymbol{B}_k^\top$ represent the corresponding discrete *coboundary operators*, which are adjoint to the boundary operators under the standard Euclidean inner product. When acting on simplicial cochains, these operators can be interpreted as discrete analogues of differential operators. In particular, $\boldsymbol{B}_1^\top$, $\boldsymbol{B}_2^\top$, and $-\boldsymbol{B}_1$ correspond to the gradient, curl, and divergence operators, respectively [Lim, 2020].

3.1 Discrete Hodge Laplacian

In Euclidean space, the Laplace operator acting on a scalar function can be expressed as the divergence of its gradient; informally, it measures how the value of the function at a point differs from its values in the surrounding neighbourhood. On graphs, an analogous operator is given by the *graph Laplacian*

$$\boldsymbol{L}_0 \coloneqq \boldsymbol{B}_1 \boldsymbol{B}_1^\top \in \mathbb{Z}^{N_0 \times N_0}, \tag{4}$$

where $\boldsymbol{B}_1$ is commonly called the graph incidence matrix. This construction extends to simplicial complexes through the *Hodge Laplacian* [Lim, 2020],

$$\boldsymbol{L}_k \coloneqq \boldsymbol{L}_k^{\text{low}} + \boldsymbol{L}_k^{\text{up}} \in \mathbb{Z}^{N_k \times N_k}, \tag{5}$$

where $\boldsymbol{L}_k^{\text{low}} \coloneqq \boldsymbol{B}_k^\top \boldsymbol{B}_k$ is the *lower Laplacian* and $\boldsymbol{L}_k^{\text{up}} \coloneqq \boldsymbol{B}_{k+1}\boldsymbol{B}_{k+1}^\top$ is the *upper Laplacian*. The lower Laplacian encodes interactions between k-simplices through $(k-1)$-simplices, whereas the upper Laplacian captures interactions between k-simplices through $(k+1)$-simplices.

The Hodge Laplacian is not the only Laplace-type operator; another example is the connection Laplacian [Petersen, 2016]. However, the continuous Hodge Laplacian and its discrete counterpart have the particular advantage of inducing the Hodge decomposition, which we discuss next.

3.2 Hodge decomposition

The *Helmholtz decomposition*, often referred to as the fundamental theorem of vector calculus, states that, under suitable regularity and boundary conditions, a vector field can be decomposed into curl-free, divergence-free, and harmonic components. The curl-free component contains no rotational part (no vortices), the divergence-free component contains no sources or sinks, and the harmonic component is both curl-free and divergence-free. The discrete *Hodge decomposition* provides an analogous decomposition for signals supported on simplicial complexes [Lim, 2020], decomposing the attribute space into a direct sum of three subspaces that are orthogonal with respect to the standard Euclidean inner product,

$$\mathbb{R}^{N_k} = \operatorname{im}\big(\boldsymbol{B}_k^\top\big) \oplus \operatorname{im}\big(\boldsymbol{B}_{k+1}\big) \oplus \ker\big(\boldsymbol{L}_k\big). \tag{6}$$

The subspaces $\operatorname{im}(\boldsymbol{B}_k^\top)$, $\operatorname{im}(\boldsymbol{B}_{k+1})$, and $\ker(\boldsymbol{L}_k)$ are referred to as the *exact*, *co-exact*, and *harmonic* subspaces, respectively (see Figure 2). Collectively, these are called the *Hodge subspaces*.



(a) Divergence-free. An edge signal is divergence-free if, at each vertex, the signed sum of the incident edge values is zero.

(b) Curl-free. An edge signal is curl-free if, for each triangle, the signed sum of its edge values is zero.

Figure 2: Edge signals in the exact subspace are curl-free, whereas those in the co-exact subspace are divergence-free. Signals in the harmonic subspace are both curl-free and divergence-free.

The Hodge decomposition also induces a partition of the eigendecomposition $\boldsymbol{L}_k = \boldsymbol{U}_k\boldsymbol{\Lambda}_k\boldsymbol{U}_k^\top$ according to the Hodge subspaces:

$$\boldsymbol{U}_k = \big[\boldsymbol{U}_{ke}\ \boldsymbol{U}_{kc}\ \boldsymbol{U}_{kh}\big] \in \mathbb{R}^{N_k \times (N_{ke}+N_{kc}+N_{kh})}, \tag{7}$$

$$\boldsymbol{\Lambda}_k = \operatorname{diag}\big(\boldsymbol{\Lambda}_{ke}, \boldsymbol{\Lambda}_{kc}, \boldsymbol{\Lambda}_{kh}\big) \in \mathbb{R}^{N_k \times (N_{ke}+N_{kc}+N_{kh})}, \tag{8}$$

where N_{ke}, N_{kc}, and N_{kh} denote the dimensions of the exact, co-exact, and harmonic subspaces, respectively, with $N_k = N_{ke} + N_{kc} + N_{kh}$. Furthermore, $(\boldsymbol{\Lambda}_{ke}, \boldsymbol{U}_{ke})$ and $(\boldsymbol{\Lambda}_{kc}, \boldsymbol{U}_{kc})$ are the non-zero eigenpairs associated with the lower and upper Laplacians $\boldsymbol{L}_k^{\text{low}}$ and $\boldsymbol{L}_k^{\text{up}}$, respectively, while $(\boldsymbol{\Lambda}_{kh}, \boldsymbol{U}_{kh})$ are the zero eigenpairs of the Hodge Laplacian $\boldsymbol{L}_k$.

The value of this decomposition is that the three components carry different information and can be treated separately. In particular, the dimension of the harmonic subspace equals the number of k-dimensional topological holes in the simplicial complex. Moreover, a signal may be dominated by one Hodge component; for example, a vortex-dominated flow lies mostly in the co-exact subspace. Allowing a model to weight these components separately can therefore improve its predictive flexibility.

When $k = 0$, the lower Laplacian $\boldsymbol{L}_0^{\text{low}}$ is zero and the exact subspace vanishes, reflecting the absence of simplices of dimension below the vertices. Similarly, when $k = K$, the upper Laplacian $\boldsymbol{L}_K^{\text{up}}$ is zero and the co-exact subspace vanishes because there are no $(K+1)$-simplices. For a graph, viewed as a simplicial 1-complex, this means that vertex attributes have no exact component and edge attributes have no co-exact component. A co-exact component for edge signals can be introduced by augmenting the graph with 2-simplices, for example, by filling its 3-cliques to obtain a simplicial 2-complex.

4 Hodgelet representations for SC prediction with GPs

We are now ready to introduce a representation of attributed SC inputs that exploits both their attributes and topology. In Section 3, we saw that the Hodge Laplacian is a discrete analogue of a differential operator acting on the attribute space that captures the structure of an SC, including interactions between simplices of different dimensions. As we shall see, the Hodge Laplacian is also naturally related to the notion of the Fourier transform.

4.1 Hodgelet transforms

The *Fourier transform* of a signal supported on the k-simplices is obtained by projecting the signal onto the eigenbasis of the Hodge Laplacian $\boldsymbol{L}_k$. This provides a representation of the signal in the spectral domain, making it possible to identify and manipulate specific frequencies associated with the eigenvalues of the Hodge Laplacian. Such a representation can reveal patterns that may not be apparent when examining the original signal. However, the Fourier transform is not localised in space, which limits its ability to capture spatially localised patterns.

To address this limitation, wavelet transforms provide simultaneous localisation in the spatial and spectral domains. Classical wavelet constructions can be adapted to signals supported on graphs [Hammond et al., 2011] and simplicial complexes [Roddenberry et al., 2022]. In our setting, this is achieved by first defining the *wavelet filter*

$$w_k(\lambda) \coloneqq \eta_k \, a_k\big(\alpha_k \lambda\big) + \sum_{l=1}^{J_k} \eta_{kl} \, b_k\big(\beta_{kl} \lambda\big), \qquad P_k \coloneqq \big\{\eta_k, \eta_{k1}, \ldots, \eta_{kJ_k}, \alpha_k, \beta_{k1}, \ldots, \beta_{kJ_k}\big\}, \tag{9}$$

where $a_k : \mathbb{R} \to \mathbb{R}$ is a scaling function, $b_k : \mathbb{R} \to \mathbb{R}$ is a wavelet function, η_k and η_{kl} are learnable filter amplitudes, J_k is the number of band-pass scales, and P_k is the corresponding collection of wavelet parameters. The wavelet function acts as a band-pass filter that captures information at intermediate to high frequencies, whereas the scaling function acts as a low-pass filter that captures low-frequency information.

Rather than applying a single wavelet filter to the full spectrum $\boldsymbol{\Lambda}_k$, we apply separate filters to the exact, co-exact, and harmonic components $\boldsymbol{\Lambda}_{ke}$, $\boldsymbol{\Lambda}_{kc}$, and $\boldsymbol{\Lambda}_{kh}$, respectively. We denote these filters by w_{ke}, w_{kc}, and w_{kh}, with corresponding parameter sets P_{ke}, P_{kc}, and P_{kh}. The k-simplex wavelet transforms [Roddenberry et al., 2022] associated with the three Hodge components, called the *Hodgelet transforms*, are then defined as the matrices

$$\mathbf{W}_{P_{ke}} \coloneqq \boldsymbol{U}_{ke} w_{ke}\big(\boldsymbol{\Lambda}_{ke}\big) \boldsymbol{U}_{ke}^\top \in \mathbb{R}^{N_k \times N_k}, \tag{10}$$

$$\mathbf{W}_{P_{kc}} \coloneqq \boldsymbol{U}_{kc} w_{kc}\big(\boldsymbol{\Lambda}_{kc}\big) \boldsymbol{U}_{kc}^\top \in \mathbb{R}^{N_k \times N_k}, \tag{11}$$

$$\mathbf{W}_{P_{kh}} \coloneqq \boldsymbol{U}_{kh} w_{kh}\big(\boldsymbol{\Lambda}_{kh}\big) \boldsymbol{U}_{kh}^\top \in \mathbb{R}^{N_k \times N_k}. \tag{12}$$

The wavelet filters $w_{k\bullet}$ are applied element-wise to the eigenvalues contained in the diagonal matrices $\boldsymbol{\Lambda}_{k\bullet}$. Note that since $\boldsymbol{\Lambda}_{kh} = \mathbf{0}$, the harmonic filter reduces to the constant $w_{kh}(0)$, which depends on the filter amplitudes but not on the scale parameters.

When applied to the signal $\boldsymbol{x}_k$, these wavelet transforms yield what we call the *Hodgelet coefficients*

$$\hat{\boldsymbol{x}}_{ke} \coloneqq \mathbf{W}_{P_{ke}} \boldsymbol{x}_k \in \mathbb{R}^{N_k}, \qquad \hat{\boldsymbol{x}}_{kc} \coloneqq \mathbf{W}_{P_{kc}} \boldsymbol{x}_k \in \mathbb{R}^{N_k}, \qquad \hat{\boldsymbol{x}}_{kh} \coloneqq \mathbf{W}_{P_{kh}} \boldsymbol{x}_k \in \mathbb{R}^{N_k}. \tag{13}$$

These coefficients are natural candidates for representing attributed simplicial complexes, as they capture information from both the attributes and the underlying structure of the complex. When the attributes are simplicial cochains, the Hodgelet coefficients are equivariant under simplex relabellings and changes of orientation, meaning that they transform consistently with these operations rather than remaining unchanged. For attributes that are not simplicial cochains, the coefficients remain equivariant under simplex relabellings but not, in general, under changes of orientation. We aggregate the coefficients to obtain fixed-dimensional representations, whose invariance properties depend on the choice of aggregation function.

4.2 Hodgelet representations

We pool the Hodgelet coefficients in Equation (13) using an *aggregation function*, similarly to the aggregation operations used in message-passing neural networks [Gilmer et al., 2017]. Ideally, this

function should be invariant under permutations and sign changes of the coefficients, so that, for cochains, the resulting representation is invariant under simplex relabellings and changes of orientation. We also require the aggregation function to output a scalar value. As a simple example, we consider the *energy*, given by the squared 2-norm,

$$A_{ke} \coloneqq \left\|\hat{\boldsymbol{x}}_{ke}\right\|^2, \quad A_{kc} \coloneqq \left\|\hat{\boldsymbol{x}}_{kc}\right\|^2, \quad A_{kh} \coloneqq \left\|\hat{\boldsymbol{x}}_{kh}\right\|^2. \tag{14}$$

The squared 2-norm is invariant under permutations and sign changes of the coefficients, thus satisfying the desired invariance properties. Other aggregation functions may also be considered, such as the sum, minimum, and maximum, or weighted combinations of these and the energy (the *mixture*), although they are not generally invariant under sign changes of the coefficients. For example, in settings where the net value of the signal is important for prediction, the sum may provide a more appropriate aggregation than the energy.

Non-cochain attributes. If the attributes do not change sign under orientation reversal (e.g., bond types), a change of orientation, given by a diagonal sign matrix $\boldsymbol{S} \in \mathbb{R}^{N_k \times N_k}$, maps the coefficients $\mathbf{W}_{P_k\bullet}\boldsymbol{x}_k$ to $\boldsymbol{S}\mathbf{W}_{P_k\bullet}\boldsymbol{S}\boldsymbol{x}_k$, and the aggregations above then generally depend on the arbitrary orientation (e.g., on the atom numbering of a molecule). For such attributes, we replace the sum and the energy by their averages over all orientations. These two *orientation-averaged aggregations*, the orientation-averaged sum and energy, have the closed forms

$$\begin{aligned} A_{k\bullet}^{\text{sum}} &\coloneqq \mathbb{E}_{\boldsymbol{S}}\left[\mathbf{1}^\top \boldsymbol{S}\mathbf{W}_{P_k\bullet}\boldsymbol{S}\boldsymbol{x}_k\right] = \textstyle\sum_{i=1}^{N_k} (\mathbf{W}_{P_k\bullet})_{ii}\,(\boldsymbol{x}_k)_i, \\ A_{k\bullet}^{\text{energy}} &\coloneqq \mathbb{E}_{\boldsymbol{S}}\left[\|\boldsymbol{S}\mathbf{W}_{P_k\bullet}\boldsymbol{S}\boldsymbol{x}_k\|^2\right] = \textstyle\sum_{i=1}^{N_k} \left(\mathbf{W}_{P_k\bullet}^2\right)_{ii}(\boldsymbol{x}_k)_i^2, \end{aligned} \tag{15}$$

which are exactly invariant under simplex relabellings and changes of orientation (see Appendix A). For such attributes, each wavelet filter contributes both values to the Hodgelet representation below, which then has twice as many entries.

Next, for each $0 \le k \le K$, we form the *Hodgelet representations* by concatenating the aggregated coefficients obtained from F_k wavelet filters:

$$\boldsymbol{r}_{k\bullet} \coloneqq \left[A_{k\bullet}^{(1)} \;\cdots\; A_{k\bullet}^{(F_k)}\right]^\top \in \mathbb{R}^{F_k}, \qquad \bullet \in \{e, c, h\}. \tag{16}$$

Using multiple wavelet filters rather than a single one enables the representation to capture information from the simplicial complex signal across multiple spectral resolutions.

Remark 1. The case of $D_k > 1$ attributes per k-simplex is handled by applying the same wavelet filters to each attribute dimension and stacking the resulting aggregations into Hodgelet representations of dimension $F_k D_k$ rather than F_k.

Remark 2. The most expensive step in computing the Hodgelet coefficients is the eigendecomposition of $\boldsymbol{L}_k$, which scales cubically with the number N_k of k-simplices. For large SCs, this cost can be reduced with Lanczos solvers or with polynomial approximations of the filters in $\boldsymbol{L}_k^{\text{low}}$ and $\boldsymbol{L}_k^{\text{up}}$ [Defferrard et al., 2016]; the SCs in many prediction tasks are, however, small enough for a full eigendecomposition.

4.3 Hodgelet kernel function

Recall that $\mathcal{S}$ denotes a set of attributed simplicial complexes of dimension at most K. Given the Hodgelet representations, we can now define the covariance kernel $\kappa : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ as an additive kernel, with one term for each simplex dimension and Hodge component:

$$\kappa(\mathcal{K}, \mathcal{K}') \coloneqq \sum_{k=0}^{K} \left(\sigma_{ke}^2\,\kappa_{ke}(\boldsymbol{r}_{ke}, \boldsymbol{r}'_{ke}) + \sigma_{kc}^2\,\kappa_{kc}(\boldsymbol{r}_{kc}, \boldsymbol{r}'_{kc}) + \sigma_{kh}^2\,\kappa_{kh}(\boldsymbol{r}_{kh}, \boldsymbol{r}'_{kh})\right), \tag{17}$$

for any simplicial complexes $\mathcal{K}, \mathcal{K}' \in \mathcal{S}$, where $\kappa_{k\bullet} : \mathbb{R}^{F_k} \times \mathbb{R}^{F_k} \to \mathbb{R}$, for $\bullet \in \{e, c, h\}$, are standard kernels with unit amplitude, such as the RBF kernel or kernels from the Matérn family, and the output scales $\sigma_{ke}^2, \sigma_{kc}^2, \sigma_{kh}^2 > 0$ weight the Hodge components; a scale close to zero effectively switches its component off. The sum over the Hodge components mirrors the Hodge decomposition (6), which writes a signal as the sum of its exact, co-exact, and harmonic parts. Other combinations of the component kernels are possible, such as products across simplex dimensions. The kernel

hyperparameters (the output scales $\sigma^2_{k\bullet}$ and the parameters of the kernels $\kappa_{k\bullet}$) are optimised jointly with the wavelet parameters $P_{k\bullet}$ of the F_k filters, which trains the model end-to-end (the full pipeline is summarised in Appendix C.3). As with standard GPs, exact training and inference have cubic time complexity $\mathcal{O}(m^3)$ in the number m of training SCs, although sparse variational GPs with M inducing points, which we use in all experiments, reduce it to $\mathcal{O}(m_b M^2 + M^3)$ per mini-batch of size m_b [Titsias, 2009, Hensman et al., 2013]. For a fixed dataset, the eigendecompositions required to construct the Hodgelet representations constitute a one-off preprocessing cost and can be computed offline.

Using separate kernels for each Hodge component provides greater modelling flexibility than using a single kernel and can improve predictive performance for edge flows [Yang et al., 2024]. In addition, our construction supports continuous, multidimensional simplex attributes and SCs of varying sizes, in contrast to many conventional graph-kernel methods.

5 Experiments

We consider three series of experiments (vector fields, molecules, and power grids), covering binary and multiclass classification and regression. Our aim is to demonstrate that (1) adding edge information can improve the performance of our models, (2) the Hodge decomposition offers a natural way of increasing the flexibility of our SC representations, and (3) when edge attributes are present, it is preferable to treat them as edge attributes rather than as vertex attributes on the corresponding line graph.

As baselines, we consider the graph neural network (GNN) architectures GCN [Kipf and Welling, 2017], GAT [Veličković et al., 2018], GIN [Xu et al., 2019, Hu et al., 2020], and Graph Transformer [Shi et al., 2021]; GCN uses only vertex attributes, while GAT, GIN, and Transformer use both vertex and edge attributes. Appendix D adds simplicial (SNN, SCNN) and Weisfeiler–Lehman kernel (WL-SVM, WL-GP) baselines on Div-curl-free, Vortices, and MUTAG. We denote by WTGP and HTGP the GP models whose SC representations are based on the wavelet transform and on the Hodgelet transforms, respectively; the latter applies wavelet transforms to each Hodge component separately. The labels "vertex", "edge", and "hybrid" indicate whether a model uses the vertex attributes, the edge attributes, or both; for example, WTGP (vertex) is the GP model of Opolka et al. [2023]. We evaluate each model over five independent repetitions (independently generated datasets, random subsamples, or random data splits). Within each repetition, we run five-fold cross-validation and compute all metrics from the pooled out-of-fold predictions; we report the mean and standard error over the repetitions. The aggregation function is fixed per benchmark (see Appendix C); the negative log predictive density (NLPD) and expected calibration error (ECE) are reported in Appendix D.

5.1 Vector field classification

We first conduct experiments on two synthetic datasets, which we refer to as Div-curl-free and Vortices. Both contain 100 complexes, obtained by discretising random vector fields into edge signals on triangular meshes via the de Rham map [Desbrun et al., 2008] (see Appendix C.2). The tasks are to determine whether a vector field is predominantly divergence-free or curl-free (Div-curl-free), or whether the net circulation of a vortex-dominated fluid flow is positive or negative, i.e., anticlockwise or clockwise (Vortices). Each task is performed at different mesh resolutions, given by the number of vertices, ranging from 10 to 150. Within a generated Vortices dataset, all complexes share one mesh, and hence one edge orientation (see Appendix C.2). We also construct line-graph counterparts on which WTGP (vertex) and HTGP (vertex) are trained.

Figure 3 displays the accuracy of the GP models at different mesh resolutions. On both datasets, converting the edge-attributed complexes into line graphs with vertex attributes lowers the accuracy, highlighting the importance of preserving edge-level structure. On Div-curl-free, the line-graph models are close to chance level from 25 vertices on (0.49 to 0.55), whereas the edge models stay well above it; a likely reason is that line graphs lose the edge orientations, and with them the signed sums at vertices and around triangles that define divergence and curl. HTGP (edge) is the best-performing model on Div-curl-free at every resolution, ahead of WTGP (edge) by 0.06 to 0.10. On Vortices, HTGP (edge) and WTGP (edge) attain accuracies of 0.75 to 0.93 and 0.71 to 0.91, both well ahead of the line-graph models (0.57 to 0.63).

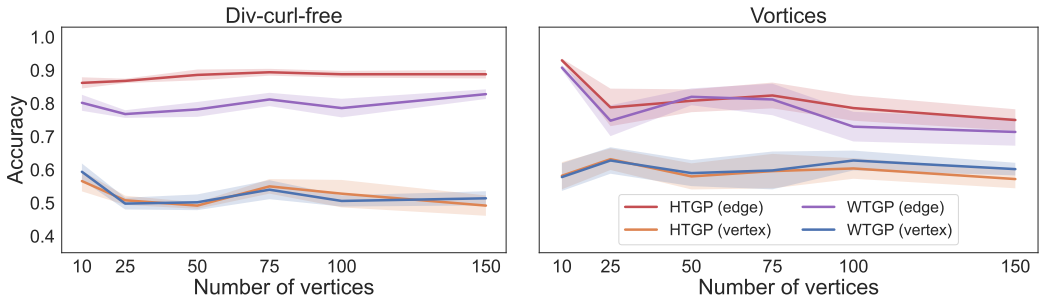


Figure 3: Accuracy of the GP models versus the mesh resolution (left: Div-curl-free, energy aggregation; right: Vortices, sum aggregation). Shaded regions indicate the standard error over five independently generated datasets of 100 complexes.

Table 1: Mean and standard error over five repetitions with different random splits. Colours indicate the best, second-best, and third-best result for each dataset.

	Accuracy (↑)		Mean squared error (↓)	
	MUTAG	AIDS ($\times 10^{-2}$)	FreeSolv	ESOL
GCN (vertex)	0.790 ± 0.014	99.60 ± 0.04	1.86 ± 0.11	0.47 ± 0.01
GAT (hybrid)	0.763 ± 0.009	99.59 ± 0.02	2.19 ± 0.14	$\mathbf{0.45} \pm 0.01$
GIN (hybrid)	0.802 ± 0.016	99.54 ± 0.02	2.06 ± 0.14	1.02 ± 0.08
Transformer (hybrid)	0.789 ± 0.016	99.55 ± 0.04	1.99 ± 0.23	0.48 ± 0.01
WTGP (hybrid)	$\mathbf{0.863} \pm 0.010$	$\mathbf{99.61} \pm 0.04$	$\mathbf{1.20} \pm 0.04$	$\mathbf{0.41} \pm 0.01$
WTGP (vertex)	0.819 ± 0.006	99.14 ± 0.05	$\mathbf{1.22} \pm 0.05$	$\mathbf{0.41} \pm 0.01$
WTGP (edge)	$\mathbf{0.878} \pm 0.007$	99.58 ± 0.02	9.22 ± 0.16	1.62 ± 0.02
HTGP (hybrid)	$\mathbf{0.878} \pm 0.009$	$\mathbf{99.64} \pm 0.01$	$\mathbf{1.26} \pm 0.02$	$\mathbf{0.43} \pm 0.01$
HTGP (vertex)	0.815 ± 0.006	99.11 ± 0.03	1.32 ± 0.02	$\mathbf{0.45} \pm 0.01$
HTGP (edge)	$\mathbf{0.870} \pm 0.003$	$\mathbf{99.63} \pm 0.03$	8.75 ± 0.11	1.59 ± 0.01

5.2 TUDataset and MoleculeNet

The second series of datasets is taken from the standard molecular graph benchmarks TUDataset [Morris et al., 2020] and MoleculeNet [Wu et al., 2018]. The vertex and edge attributes are one-hot encodings of the atom and bond types (all atom and bond features for MoleculeNet). Unlike flows, bond attributes do not change sign under orientation reversal, so aggregations of their Hodgelet coefficients generally depend on the edge orientations, which follow the atom numbering; we therefore use the orientation-averaged sum and energy (15), which do not depend on these orientations. As for all graph benchmarks in this paper, each graph is lifted to a simplicial 2-complex by filling every 3-clique (see Appendix A). MUTAG and AIDS are binary classification tasks, while FreeSolv and ESOL are regression tasks. We summarise our results in Table 1. For MUTAG, the four GP models that use the edge attributes are the best-performing models, ahead of all GNN baselines (at most 0.802) and of their vertex-only counterparts (by four to six points). For AIDS, all models are close to the accuracy ceiling. For FreeSolv and ESOL, WTGP (hybrid), WTGP (vertex), and HTGP (hybrid) are the three best-performing models and all outperform the GNN baselines. On both regression datasets, the bond attributes are less informative than the atom attributes, and the additive kernel compensates for this in the hybrid models, which are on par with or better than their vertex-only counterparts. On these molecular benchmarks, whose bond attributes are not flows, HTGP and WTGP perform similarly. On the flows of the vector fields and power grids, HTGP (edge) is generally ahead of WTGP (edge). Overall, the GP models are the best-performing models or on par with the best GNN on each dataset. On MUTAG, they are also better calibrated: every GP model has a lower NLPD and ECE than every GNN (e.g., an NLPD of 0.296 for HTGP (hybrid), against 0.452 for GIN, the best GNN; see Appendix D).

5.3 PowerGraph

In the last series of experiments, we consider prediction tasks in electricity networks using the ieee24 data from the PowerGraph benchmark [Varbella et al., 2024]. We subsample this real-world dataset to the small-data regime (see Appendix C.1). The graphs represent power grids at different pre-

outage operating conditions, with electrical buses as vertices and transmission lines and transformers as edges. The vertex attributes are the three bus features provided by PowerGraph (the net active power, the net apparent power, and the voltage magnitude). The edge attributes consist of the active and reactive power flows, the branch reactance, and the line rating. Since all graphs share the same grid, a single edge orientation is used for every sample. The three tasks are to determine whether (binary classification) and to what extent (multiclass classification and regression) a power grid is stable. In Table 2, HTGP (hybrid) and HTGP (edge) are the two best-performing models on ieee24-binary, ahead of the best GNN, and the four GP models that use the edge attributes are also ahead of all GNN baselines on ieee24-multiclass. On ieee24-regression, HTGP (hybrid) and HTGP (edge) are again the two best-performing models. The edge attributes clearly improve performance: on both classification tasks, the GP models that use them outperform their vertex-only counterparts, by three to nine points on ieee24-binary and by fourteen to sixteen points on ieee24-multiclass. The Hodge decomposition also helps with these power flows on ieee24-binary and ieee24-regression, where HTGP (hybrid) and HTGP (edge) outperform their WTGP counterparts; on ieee24-multiclass, the HTGP and WTGP models are on par. HTGP (edge) also attains the lowest ECE on both classification tasks and the lowest NLPD, tied with WTGP (edge) on ieee24-multiclass (see Appendix D).

Table 2: Mean and standard error over five independent subsamples. Colours indicate the best, second-best, and third-best result for each dataset.

	Accuracy (↑)		Mean squared error (↓)
	ieee24-binary	ieee24-multiclass	ieee24-regression ($\times 10^{-4}$)
GCN (vertex)	0.767 ± 0.010	0.689 ± 0.026	4.21 ± 0.31
GAT (hybrid)	0.859 ± 0.010	0.872 ± 0.009	3.77 ± 0.26
GIN (hybrid)	$\mathbf{0.885} \pm 0.014$	0.876 ± 0.010	2.58 ± 0.19
Transformer (hybrid)	0.879 ± 0.013	0.889 ± 0.004	3.48 ± 0.32
WTGP (hybrid)	0.879 ± 0.013	0.906 ± 0.004	$\mathbf{2.19} \pm 0.10$
WTGP (vertex)	0.811 ± 0.006	0.766 ± 0.011	3.05 ± 0.24
WTGP (edge)	0.843 ± 0.025	$\mathbf{0.921} \pm 0.003$	4.79 ± 0.81
HTGP (hybrid)	$\mathbf{0.906} \pm 0.010$	$\mathbf{0.911} \pm 0.008$	$\mathbf{1.61} \pm 0.20$
HTGP (vertex)	0.813 ± 0.009	0.759 ± 0.017	3.16 ± 0.21
HTGP (edge)	$\mathbf{0.902} \pm 0.007$	$\mathbf{0.919} \pm 0.007$	$\mathbf{1.84} \pm 0.24$

6 Conclusion

In this paper, we propose a representation of attributed SCs and graphs for prediction tasks, which, combined with GPs, gives a robust and interpretable method that achieves competitive performance and well-calibrated predictions across a variety of datasets. Our Hodgelet representations leverage both the wavelet transform of the attributes, to capture multi-scale spectral and spatial information, and the Hodge decomposition, to separate the signal into exact, co-exact, and harmonic components for finer control. In our experiments, the GP models match or outperform the GNN baselines on the molecular and power-grid benchmarks; keeping the edge attributes on the edges, rather than moving them to a line graph, substantially improves accuracy on vector fields; and separating the Hodge components helps most on Div-curl-free and the power grids. On the orientability of triangulated surfaces, a purely topological task, the GP models are almost perfect, well ahead of the neural baselines (see Appendix D.10). In future work, we intend to incorporate the geometry of the SC, which has been shown to be useful for molecular prediction tasks [Fang et al., 2022], for example through its curvature [Ni et al., 2019], and to adapt our approach to prediction tasks at the level of vertices or edges.

Acknowledgments and Disclosure of Funding

MA is supported by a Mathematical Sciences Doctoral Training Partnership held by Prof. Helen Wilson, funded by the Engineering and Physical Sciences Research Council (EPSRC), under Project Reference EP/W523835/1. ST is supported by a Department of Defense Vannevar Bush Faculty Fellowship held by Prof. Andrew Stuart, and by the SciAI Center, funded by the Office of Naval Research (ONR), under Grant Number N00014-23-1-2729.

References

Mathieu Alain, So Takao, Brooks Paige, and Marc P Deisenroth. Gaussian processes on cellular complexes. In *International Conference on Machine Learning*, 2024a. (cited on pages 2, 15, and 16)

Mathieu Alain, So Takao, Bastian Rieck, Xiaowen Dong, and Emmanuel Noutahi. Graph Classification Gaussian Processes via Hodgelet Spectral Features. In *Advances in Neural Information Processing Systems (NeurIPS) 2024 Workshop Workshop on Bayesian Decision-making and Uncertainty (BDU)*, 2024b. URL `https://arxiv.org/pdf/2410.10546`. (cited on page 2)

Federica Baccini, Filippo Geraci, and Ginestra Bianconi. Weighted simplicial complexes and their representation power of higher-order network data and topology. *Physical Review E*, 2022. (cited on page 15)

Rubén Ballester, Ernst Röell, Daniel Bin Schmid, Mathieu Alain, Sergio Escalera, Carles Casacuberta, and Bastian Rieck. MANTRA: The manifold triangulations assemblage. In *International Conference on Learning Representations*, 2025. (cited on pages 1 and 29)

Sergio Barbarossa and Stefania Sardellitti. Topological signal processing over simplicial complexes. *IEEE Transactions on Signal Processing*, 2020. (cited on pages 2 and 3)

Claudio Battiloro, Paolo Di Lorenzo, and Sergio Barbarossa. Topological Slepians: Maximally localized representations of signals over simplicial complexes. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023. (cited on page 2)

J Thomas Beale and Andrew Majda. High order accurate vortex methods with explicit velocity kernels. *Journal of Computational Physics*, 58(2):188–208, 1985. (cited on page 19)

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Liò, and Michael Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1026–1037. PMLR, 2021. (cited on page 1)

Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *Foundations and Trends in Machine Learning*, 2020. (cited on page 1)

Michael Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv, 2021. (cited on page 1)

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Conference on Neural Information Processing Systems*, 2016. (cited on page 7)

Mathieu Desbrun, Eva Kanso, and Yiying Tong. *Discrete Differential Forms for Computational Modeling*, pages 287–324. Birkhäuser Basel, 2008. (cited on pages 8 and 18)

Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial neural networks. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020. (cited on page 26)

Xiaomin Fang, Lihang Liu, Jieqiong Lei, Donglong He, Shanzhuo Zhang, Jingbo Zhou, Fan Wang, Hua Wu, and Haifeng Wang. Geometry-enhanced molecular representation learning for property prediction. *Nature Machine Intelligence*, 4(2):127–134, 2022. (cited on page 10)

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *AAAI Conference on Artificial Intelligence*, 2019. (cited on page 15)

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. (cited on page 17)

Dominik Fuchsgruber, Tim Postuvan, Stephan Günnemann, and Simon Geisler. Graph neural networks for edge signals: Orientation equivariance and invariance. In *International Conference on Learning Representations*, 2025. (cited on page 16)

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018. (cited on page 17)

Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017. (cited on page 22)

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. (cited on page 6)

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017. (cited on page 21)

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 2011. (cited on pages 2 and 6)

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intelligence*, 2013. (cited on pages 8 and 17)

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*, 2020. (cited on pages 8 and 21)

Elvin Isufi, Geert Leus, Baltasar Beferull-Lozano, Sergio Barbarossa, and Paolo Di Lorenzo. Topological signal processing and learning: Recent advances and future challenges. *Signal Processing*, 2025. (cited on page 2)

Thomas Kipf and Max Welling. Semi-supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017. (cited on page 8)

Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 2020. (cited on page 1)

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, 2018. (cited on page 23)

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. (cited on page 1)

Lek-Heng Lim. Hodge Laplacians on graphs. *SIAM Review*, 2020. (cited on pages 4 and 5)

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 2020. (cited on page 1)

Noël Malod-Dognin and Nataša Pržulj. Functional geometry of protein interactomes. *Bioinformatics*, 35(19):3727–3734, 2019. doi: 10.1093/bioinformatics/btz146. (cited on page 1)

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020. (cited on page 9)

Paul K Newton. *The N-Vortex Problem: Analytical Techniques*. Applied Mathematical Sciences. Springer, New York, 2001. (cited on page 19)

Chien-Chun Ni, Yu-Yao Lin, Feng Luo, and Jie Gao. Community detection on networks with Ricci flow. *Scientific Reports*, 2019. (cited on page 10)

Felix Opolka, Yin-Cong Zhi, Pietro Liò, and Xiaowen Dong. Graph classification Gaussian processes via spectral features. In *Conference on Uncertainty in Artificial Intelligence*, 2023. (cited on pages 2 and 8)

Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 2009. (cited on page 17)

Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 2018. (cited on page 2)

Theodore Papamarkou, Tolga Birdal, Michael Bronstein, Gunnar Carlsson, Justin Curry, Yue Gao, Mustafa Hajij, Roland Kwitt, Pietro Liò, Paolo Di Lorenzo, Vasileios Maroulas, Nina Miolane, Farzana Nasrin, Karthikeyan Natesan Ramamurthy, Bastian Rieck, Simone Scardapane, Michael T Schaub, Petar Veličković, Bei Wang, Yusu Wang, Guo-Wei Wei, and Ghada Zamzmi. Position: Topological deep learning is the new frontier for relational learning. In *International Conference on Machine Learning*, 2024. (cited on pages 1 and 15)

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019. (cited on page 17)

Peter Petersen. *Riemannian Geometry*, volume 171 of *Graduate Texts in Mathematics*. Springer, 3 edition, 2016. (cited on page 5)

Thomas Pinder, Kathryn Turnbull, Christopher Nemeth, and David Leslie. Gaussian processes on hypergraphs. *arXiv preprint arXiv:2106.01982*, 2021. (cited on page 17)

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007. (cited on page 18)

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian processes for machine learning*. MIT Press, 2005. (cited on pages 2 and 4)

Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph neural networks for materials science and chemistry. *Communications Materials*, 2022. (cited on page 1)

T. Mitchell Roddenberry, Florian Frantzen, Michael T. Schaub, and Santiago Segarra. Hodgelets: Localized spectral representations of flows on simplicial complexes. In *International Conference on Acoustics, Speech, and Signal Processing*, 2022. (cited on pages 2, 6, and 15)

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. (cited on page 1)

François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. Text categorization as a graph classification problem. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1702–1712, 2015. doi: 10.3115/v1/P15-1164. (cited on page 1)

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler–Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011. (cited on page 26)

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *International Joint Conference on Artificial Intelligence*, 2021. (cited on page 8)

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. *International Conference on Artificial intelligence and Statistics*, 2009. (cited on pages 8 and 17)

Anna Varbella, Kenza Amara, Blazhe Gjorgiev, Mennatallah El-Assady, and Giovanni Sansavini. PowerGraph: A power grid benchmark dataset for graph neural networks. In *NeurIPS 2024 Track on Datasets and Benchmarks*, 2024. (cited on page 9)

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018. (cited on page 8)

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. (cited on page 9)

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. (cited on page 8)

Maosheng Yang, Elvin Isufi, Michael T. Schaub, and Geert Leus. Simplicial convolutional filters. *IEEE Transactions on Signal Processing*, 70:4633–4648, 2022. (cited on page 26)

Maosheng Yang, Viacheslav Borovitskiy, and Elvin Isufi. Hodge-compositional edge Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, 2024. (cited on pages 2, 8, and 15)

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019. (cited on page 1)

A Simplicial complexes

In this appendix, we provide details on simplicial complexes, which are the key objects of this work. A graph consists of a set V of vertices and a set E of edges, where each edge is a pair $e = \{v_0, v_1\}$ of distinct vertices. Graphs can carry signals on their vertices and edges, resulting in a structure that is richer than tabular data. Common examples are social networks, electrical grids (see Section 5.3), citation networks, traffic maps, collaboration networks, chemical reaction networks, vector fields (see Section 5.1), and molecules (see Section 5.2). However, a limitation of graphs is that they can only encode *dyadic interactions* between vertices through their edges. Multiple recent works have sought to overcome this limitation by using more general structures, such as simplicial complexes, cellular complexes, and hypergraphs [Feng et al., 2019, Baccini et al., 2022, Roddenberry et al., 2022, Alain et al., 2024a, Yang et al., 2024, Papamarkou et al., 2024].

Simplicial complexes. Simplicial complexes are a natural extension of graphs, built from k-*simplices*. A k-simplex is a set of $k+1$ vertices: vertices are 0-simplices, edges are 1-simplices, and a 2-simplex, also called a triangular face or a triangle, is formed by three pairwise adjacent vertices and describes a *triadic interaction* between them.

A *face* of a k-simplex is a $(k-1)$-simplex formed by a subset of k of the vertices defining the k-simplex. Then, as in Section 2.1, a *simplicial K-complex* is a collection of simplex sets $S_0, \dots, S_K$, where S_k is a set of k-simplices, such that the faces of every simplex are also simplices of the complex. In other words, the union $S_0 \cup \cdots \cup S_K$ is a set of sets, each with at most $K+1$ elements, that is closed under the operation of taking non-empty subsets. For example, a graph $G = (V, E)$ is a simplicial 1-complex with $S_0 = \{\{v\} : v \in V\}$ and $S_1 = E$, since both vertices of every edge are in V.

In this work, we are primarily concerned with simplicial 2-complexes, which consist of the sets S_0, S_1, and S_2 of vertices, edges, and triangular faces, respectively. By our definition above, these sets are required to satisfy the following closure conditions:

1. all vertices that form an edge in S_1 must be in S_0, and
2. all edges that form a triangle in S_2 must be in S_1.

Since an edge is a set of two distinct vertices and is determined by them, there are no self-loops or multi-edges. Finally, as in Section 2.1, the numbers of vertices, edges, and triangles are denoted by N_0, N_1, and N_2, respectively.

Orientation. An orientation of a simplicial complex assigns to each simplex $\{v_0, \dots, v_k\} \in S_k$, for $0 \le k \le K$, an ordering of its vertices, where orderings that differ by an even permutation are considered equivalent. We denote by $[(v_0, \dots, v_k)]$ the equivalence class of the ordering $(v_0, \dots, v_k)$, and say that the simplex is oriented as $(v_0, \dots, v_k)$. An edge $\{v_0, v_1\}$ thus has two orientations, the directions (v_0, v_1) and (v_1, v_0), and a triangle $\{v_0, v_1, v_2\}$ has the two orientations

$$[(v_0, v_1, v_2)] = \{(v_0, v_1, v_2), (v_1, v_2, v_0), (v_2, v_0, v_1)\},$$
$$[(v_2, v_1, v_0)] = \{(v_2, v_1, v_0), (v_1, v_0, v_2), (v_0, v_2, v_1)\},$$

which correspond to the two directions of traversing the triangle, clockwise and anticlockwise. An orientation is needed to define edge flows and the incidence matrices below, but its choice is arbitrary.

Given an orientation, the entry $(\boldsymbol{B}_k)_{ij}$ of the incidence matrix is ± 1 if the i-th $(k-1)$-simplex is a face of the j-th k-simplex, and 0 otherwise. The sign is positive if the orientation of the face agrees with the orientation that the k-simplex induces on it, and negative otherwise. Here, $[(v_0, \dots, v_k)]$ induces on the face without v_l the orientation $[(v_0, \dots, v_{l-1}, v_{l+1}, \dots, v_k)]$ if l is even, and the opposite orientation if l is odd. For example, the column of $\boldsymbol{B}_1$ of an edge oriented as (v_0, v_1) has the entries -1 at v_0 and 1 at v_1, and the column of $\boldsymbol{B}_2$ of a triangle oriented as (v_0, v_1, v_2) has the entries 1, -1, and 1 at the edges oriented as (v_1, v_2), (v_0, v_2), and (v_0, v_1), respectively. With this convention, $\boldsymbol{B}_k \boldsymbol{B}_{k+1} = \boldsymbol{0}$, and reversing the orientation of a k-simplex flips the sign of its column of $\boldsymbol{B}_k$ and of its row of $\boldsymbol{B}_{k+1}$. In our experiments, the orientation is induced by the vertex numbering: every simplex is oriented by increasing vertex index.

Orientation-averaged aggregations. Here, we derive the orientation-averaged sum and energy (15), which are introduced in this work, and show that they are invariant under changes of orientation. When the attributes are simplicial cochains, such as edge flows, the arbitrary choice of orientation is not an issue for aggregations that are invariant under sign changes, such as the energy, since the coefficients then change sign together with the orientation. However, many attributes, such as molecular bond types, do not change sign when the orientation of a simplex is reversed. Applying wavelet filters of the signed Hodge Laplacian to such attributes makes the Hodgelet coefficients, and therefore their aggregation, depend on the chosen orientation. Fuchsgruber et al. [2025] identify the same problem for graph neural networks acting on edge signals, and process such orientation-invariant signals with unsigned incidence matrices instead of signed ones. Unsigned incidence matrices, however, do not satisfy $\boldsymbol{B}_k\boldsymbol{B}_{k+1} = \mathbf{0}$, so that the resulting Laplacian no longer admits a Hodge decomposition into orthogonal exact, co-exact, and harmonic subspaces. Since this decomposition is central to our approach, we instead keep the signed Hodge Laplacian and average the aggregation over all possible orientations.

More precisely, consider the Hodgelet transform $\mathbf{W}_{P_{k\bullet}}$ of any one Hodge component, $\bullet \in \{e, c, h\}$, defined in Section 4.1, and note that it is symmetric. Recall from Section 4.2 that a change of orientation is given by a diagonal sign matrix $\boldsymbol{S} = \operatorname{diag}(s_1, \dots, s_{N_k})$. To average over all orientations, we take the signs $s_i \in \{-1, 1\}$ to be independent, each equal to 1 or -1 with probability $1/2$. In particular, $\mathbb{E}[s_i s_j] = \delta_{ij}$, where δ_{ij} denotes the Kronecker delta. The orientation-averaged sum is then given by

$$\mathbb{E}_{\boldsymbol{S}}\left[\mathbf{1}^\top \boldsymbol{S}\mathbf{W}_{P_{k\bullet}}\boldsymbol{S}\boldsymbol{x}_k\right] = \sum_{i,j} \mathbb{E}[s_i s_j]\,(\mathbf{W}_{P_{k\bullet}})_{ij}\,(\boldsymbol{x}_k)_j = \sum_i (\mathbf{W}_{P_{k\bullet}})_{ii}\,(\boldsymbol{x}_k)_i. \tag{18}$$

Similarly, using $\boldsymbol{S}^2 = \boldsymbol{I}$ and the symmetry of $\mathbf{W}_{P_{k\bullet}}$, the orientation-averaged energy is given by

$$\begin{aligned}\mathbb{E}_{\boldsymbol{S}}\left[\|\boldsymbol{S}\mathbf{W}_{P_{k\bullet}}\boldsymbol{S}\boldsymbol{x}_k\|^2\right] &= \mathbb{E}_{\boldsymbol{S}}\left[\boldsymbol{x}_k^\top \boldsymbol{S}\mathbf{W}_{P_{k\bullet}}^2\boldsymbol{S}\boldsymbol{x}_k\right] \\ &= \sum_{i,j} \mathbb{E}[s_i s_j]\,(\mathbf{W}_{P_{k\bullet}}^2)_{ij}\,(\boldsymbol{x}_k)_i(\boldsymbol{x}_k)_j = \sum_i (\mathbf{W}_{P_{k\bullet}}^2)_{ii}\,(\boldsymbol{x}_k)_i^2.\end{aligned} \tag{19}$$

In other words, averaging over all orientations removes every term that involves two different simplices and leaves only the diagonals of $\mathbf{W}_{P_{k\bullet}}$ and of its square.

We now show that both quantities are invariant. Consider a fixed change of orientation of the k-simplices, given by a diagonal sign matrix $\boldsymbol{D} \in \mathbb{R}^{N_k \times N_k}$ (we write $\boldsymbol{D}$ to distinguish it from the random $\boldsymbol{S}$ above), whose i-th diagonal entry is -1 if the orientation of the i-th k-simplex is reversed, and 1 otherwise. This change replaces $\boldsymbol{B}_k$ by $\boldsymbol{B}_k\boldsymbol{D}$ and $\boldsymbol{B}_{k+1}$ by $\boldsymbol{D}\boldsymbol{B}_{k+1}$, while changes of orientation of the $(k-1)$- and $(k+1)$-simplices cancel out in $\boldsymbol{L}_k$. As a result, $\boldsymbol{L}_k^{\text{low}}$ and $\boldsymbol{L}_k^{\text{up}}$ become $\boldsymbol{D}\boldsymbol{L}_k^{\text{low}}\boldsymbol{D}$ and $\boldsymbol{D}\boldsymbol{L}_k^{\text{up}}\boldsymbol{D}$, which have the same eigenvalues, with eigenvectors multiplied by $\boldsymbol{D}$, so $\mathbf{W}_{P_{k\bullet}}$ becomes $\boldsymbol{D}\mathbf{W}_{P_{k\bullet}}\boldsymbol{D}$, whose square is $\boldsymbol{D}\mathbf{W}_{P_{k\bullet}}^2\boldsymbol{D}$. Multiplying a matrix by $\boldsymbol{D}$ on both sides leaves its diagonal unchanged, so the diagonals of $\mathbf{W}_{P_{k\bullet}}$ and of its square are preserved; the attributes $\boldsymbol{x}_k$, which are not cochains, do not change either. Hence, the orientation-averaged sum and energy are invariant under changes of orientation. Similarly, a relabelling of the simplices permutes the diagonals and the attributes in the same way, which leaves both sums unchanged. Since the orientation is induced by the vertex numbering, a relabelling of the vertices combines such a permutation with a change of orientation, and both aggregations are also invariant under vertex relabellings.

Lifting graphs to simplicial complexes. The graph benchmarks (TUDataset, MoleculeNet, and ieee24) are lifted to simplicial 2-complexes: the vertices and edges of a graph are its 0- and 1-simplices, and every 3-clique, that is, every set of three pairwise adjacent vertices, is added as a 2-simplex. The lift is deterministic and commutes with vertex relabellings: isomorphic graphs yield isomorphic complexes. The triangles are found by intersecting the neighbourhoods of the two endpoints of each edge, in time at most $\mathcal{O}\big(\sum_{v \in V} \deg(v)^2\big)$. The graphs carry no attributes on the triangles, so the triangles enter our models only through $\boldsymbol{B}_2$. The meshes and MANTRA are already simplicial complexes and keep their native triangles. Table 3 reports the resulting numbers of triangles, and Table 13 the effect of removing them.

Cellular complexes and hypergraphs. Simplicial complexes can be further generalised to cellular complexes, to which most of the constructions above extend [Alain et al., 2024a]. Hypergraphs,

in which an edge can join any number of vertices, could similarly be handled using Laplacians specialised for hypergraphs [Pinder et al., 2021].

B Variational Gaussian processes

For regression with Gaussian observation noise, the conjugacy between the GP prior and the likelihood allows the posterior distribution and marginal likelihood to be computed analytically. However, for classification and other tasks involving non-Gaussian likelihoods, this conjugacy is lost, and exact Bayesian inference generally becomes intractable. Consequently, approximate inference methods are required to estimate the posterior distribution and learn the model hyperparameters.

Variational GP methods [Opper and Archambeau, 2009] address this intractability by introducing a variational distribution $q_\theta(f) \approx p(f \mid \mathcal{D})$ parameterised by θ and optimised by maximising the evidence lower bound (ELBO) on the log marginal likelihood $\log p(\boldsymbol{y})$,

$$\mathrm{ELBO}(\theta) \coloneqq \mathbb{E}_{q_\theta(f)}\left[\log p(\boldsymbol{y} \mid f)\right] - \mathrm{KL}\left(q_\theta(f) \,\middle\|\, p(f)\right), \tag{20}$$

where $\boldsymbol{y} \coloneqq (y_1, \ldots, y_m)$, $p(\boldsymbol{y} \mid f)$ is the product of the likelihoods $p(y_i \mid f(\mathcal{K}_i))$, and $\mathrm{KL}(\cdot \,\|\, \cdot)$ denotes the Kullback–Leibler divergence. The variational parameters θ, the kernel and likelihood hyperparameters, and the wavelet parameters of the Hodgelet representations can be optimised jointly by maximising the ELBO; in our experiments, the KL term is down-weighted by a factor γ (see Appendix C). This provides a principled objective for learning continuous model parameters without requiring a separate validation set for their optimisation.

In all our experiments, we use sparse variational GPs, which introduce a set of M inducing variables and reduce the cost of inference from $\mathcal{O}(m^3)$ to $\mathcal{O}(m_b M^2 + M^3)$ per mini-batch of size m_b, which is linear in m per pass over the data [Titsias, 2009, Hensman et al., 2013]; the values of M are given in Appendix C.

C Experimental details

In this appendix, we provide additional details regarding the experiments described in Section 5.

For each dataset, we perform five independent repetitions. A repetition consists of a freshly generated dataset for Div-curl-free and Vortices, a fresh random subsample for ieee24, a different random split into folds for TUDataset and MoleculeNet, or a fresh balanced subsample with fresh random triangle features for MANTRA. The reported means and standard errors are computed over these five repetitions.

We implement our models in PyTorch [Paszke et al., 2019], using GPyTorch [Gardner et al., 2018] to train the Gaussian processes and PyTorch Geometric [Fey and Lenssen, 2019] to process graph data; the Div-curl-free datasets are generated with JAX.

C.1 Preprocessing

The TUDataset graphs are used as provided. For MoleculeNet, molecules that cannot be parsed or that have fewer than three atoms are removed; the vertex attributes one-hot encode the nine atom features and the edge attributes the three bond features (bond type, stereochemistry, conjugation) computed from the SMILES strings by PyTorch Geometric, whereas for TUDataset they encode the atom and bond types. For ieee24, the transmission lines and transformers that are out of service in a sample, whose features are all zero, are removed from its graph, and parallel lines between the same pair of buses are merged by averaging their features. MANTRA has no native attributes, so we use the vertex degrees, the number of triangles incident to each edge, and eight random features on each triangle, drawn independently from the standard normal distribution and redrawn in each repetition (the GPs, SNN, and SCNN use the triangle features, and the Transformer uses the vertex and edge signals). Finally, as mentioned in Section 5.3, the ieee24 datasets have been downsampled from 21500 graphs, with a new subsample in each repetition. Table 3 summarises the sizes of all datasets except MANTRA.

Table 3: Dataset statistics: number of complexes per repetition and mean numbers of vertices, edges, and triangles per complex, after lifting every 3-clique to a 2-simplex on the graph benchmarks. For Div-curl-free and Vortices, Figure 3 uses 100 complexes per generated dataset and Appendix D uses 200 unless stated otherwise; their mesh statistics are for the meshes with $n = 100$ used in Appendix D, which have 132 vertices once the points on the sides are added (Appendix C.2); Figure 3 varies the number of vertices from 10 to 150.

Dataset	Task	Complexes	Vertices	Edges	Triangles
Div-curl-free	Binary	100 / 200	132.0	357.0	226.0
Vortices	Binary	100 / 200	132.0	357.0	226.0
MUTAG	Binary	188	17.9	19.8	0.0
AIDS	Binary	2000	15.7	16.2	0.0
FreeSolv	Regression	628	8.9	8.6	0.0
ESOL	Regression	1121	13.4	13.8	0.0
ieee24-binary	Binary	≈ 376	24.0	33.1	0.9
ieee24-multiclass	Multiclass (4 classes)	≈ 351	24.0	33.0	0.9
ieee24-regression	Regression	≈ 123	24.0	33.0	1.0

C.2 Vector field classification experiments

Here, we give more details about the vector field classification experiments (Div-curl-free and Vortices) from Section 5.1.

C.2.1 Discretising vector fields via the de Rham map

To project a vector field $\boldsymbol{X} : \mathbb{R}^2 \to \mathbb{R}^2$ onto a simplicial 2-complex, we employ the *de Rham map* [Desbrun et al., 2008], which "discretises" a vector field into edge signals. For a given oriented edge $e = (v_0, v_1)$ with endpoint coordinates $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$, we define the projection $(\boldsymbol{x}_1)_e$ of $\boldsymbol{X}$ onto e by the following integral,

$$(\boldsymbol{x}_1)_e \coloneqq \int_0^1 \boldsymbol{X}\big(\boldsymbol{p}_0 + t(\boldsymbol{p}_1 - \boldsymbol{p}_0)\big) \cdot (\boldsymbol{p}_1 - \boldsymbol{p}_0)\,\mathrm{d}t, \tag{21}$$

that is, the line integral of $\boldsymbol{X}$ along the edge. This one-dimensional integral is computed by numerical quadrature. Note that this depends on the ordering of the vertices v_0 and v_1 characterising the edge: if we flip the order, then the sign of $(\boldsymbol{x}_1)_e$ flips. Consequently, the mesh must be oriented for the edge signal $\boldsymbol{x}_1$ to be well defined. In Figure 4, we display an example of such a projection onto a regular triangular mesh. The meshes of our datasets are Delaunay triangulations of the square $[-1, 1]^2$: a mesh of n vertices, as we call it throughout, is built from the four corners and $n - 4$ uniformly random interior points, with $4(\lfloor\sqrt{n}\rfloor - 2)$ evenly spaced points added on its sides, which gives exactly $n + 4(\lfloor\sqrt{n}\rfloor - 2)$ vertices (132 for $n = 100$).

C.2.2 Div-curl-free

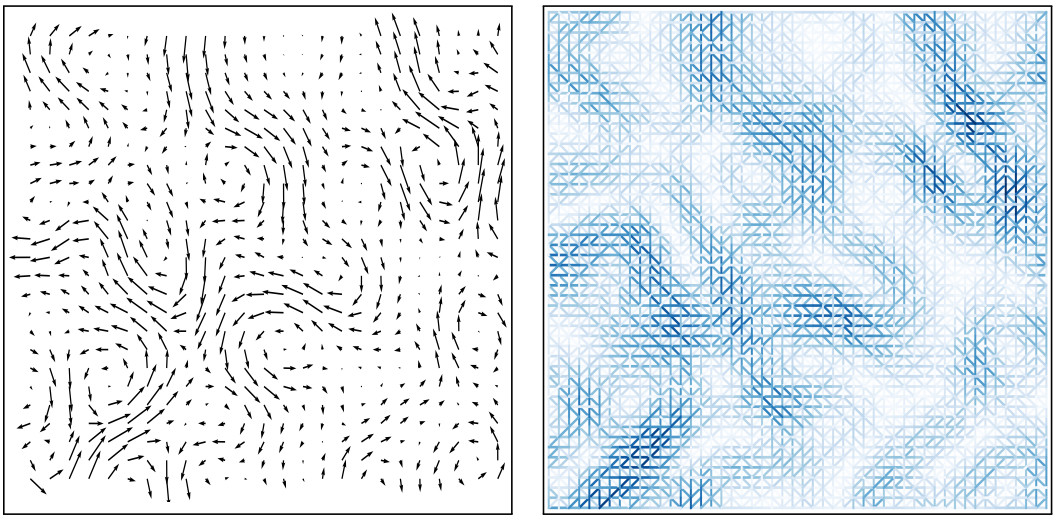
To generate a random vector field, we first sample a Gaussian process $f : \mathbb{R}^2 \to \mathbb{R}$ and then take its derivatives f_x, f_y. Since a curl-free 2D vector field is the gradient of a potential, we obtain a curl-free field by taking

$$\boldsymbol{X}_{\text{curl-free}} \coloneqq \nabla f = [f_x, f_y]^\top. \tag{22}$$

Similarly, since a divergence-free 2D vector field is the rotated gradient of a streamfunction, we obtain a divergence-free field by taking

$$\boldsymbol{X}_{\text{div-free}} \coloneqq \nabla^\perp f = [f_y, -f_x]^\top. \tag{23}$$

For the scalar field f, we use samples from a squared-exponential Gaussian process with lengthscale 0.5, drawn using its random Fourier feature approximation with 1000 cosine and 1000 sine features [Rahimi and Recht, 2007]. Figure 5 shows the derivatives f_x and f_y of such a sample and the resulting divergence-free and curl-free fields.



(a) Vector field. (b) Projection onto a triangular mesh.

Figure 4: Projection of a continuous vector field onto a regular triangular mesh by applying the de Rham map.

Our data consist of mostly divergence-free or mostly curl-free vector fields, which are generated by considering the linear combination

$$\boldsymbol{X} \coloneqq \rho \boldsymbol{X}_{\text{div-free}} + (1-\rho)\boldsymbol{X}_{\text{curl-free}}, \tag{24}$$

where ρ is drawn uniformly from $[0.15, 0.40]$ or from $[0.60, 0.85]$, with equal probability, and $\boldsymbol{X}_{\text{div-free}}$ and $\boldsymbol{X}_{\text{curl-free}}$ are obtained from the same GP sample f as above. The edge signal of a complex is the de Rham map of $\boldsymbol{X}$ on its mesh, with independent Gaussian noise of standard deviation 0.5 added to the value on each edge. The randomness of the dataset then comes from f, ρ, the noise, and the mesh, which is new for every complex. The label indicates whether $\boldsymbol{X}$ is mostly curl-free ($\rho < 0.5$) or mostly divergence-free ($\rho > 0.5$).

C.2.3 Vortices

We generate vortex-dominated flows with the point-vortex model of fluid flow [Newton, 2001]: N_v point vortices with strengths $\Gamma_1, \dots, \Gamma_{N_\text{v}}$ at locations $\boldsymbol{p}_1, \dots, \boldsymbol{p}_{N_\text{v}}$. To avoid singular velocities at the vortex locations, we use vortex-blob regularisation [Beale and Majda, 1985] with smoothing parameter $\delta = 0.1$, which gives the *streamfunction*

$$\psi_\delta(\boldsymbol{p}) \coloneqq -\sum_{n=1}^{N_\text{v}} \frac{\Gamma_n}{4\pi} \left[\log\left(\|\boldsymbol{p} - \boldsymbol{p}_n\|^2\right) - \text{Ei}(-\|\boldsymbol{p} - \boldsymbol{p}_n\|^2/\delta^2)\right], \tag{25}$$

where Ei denotes the exponential integral. The velocity field at a point $\boldsymbol{p} = (x, y)$ is

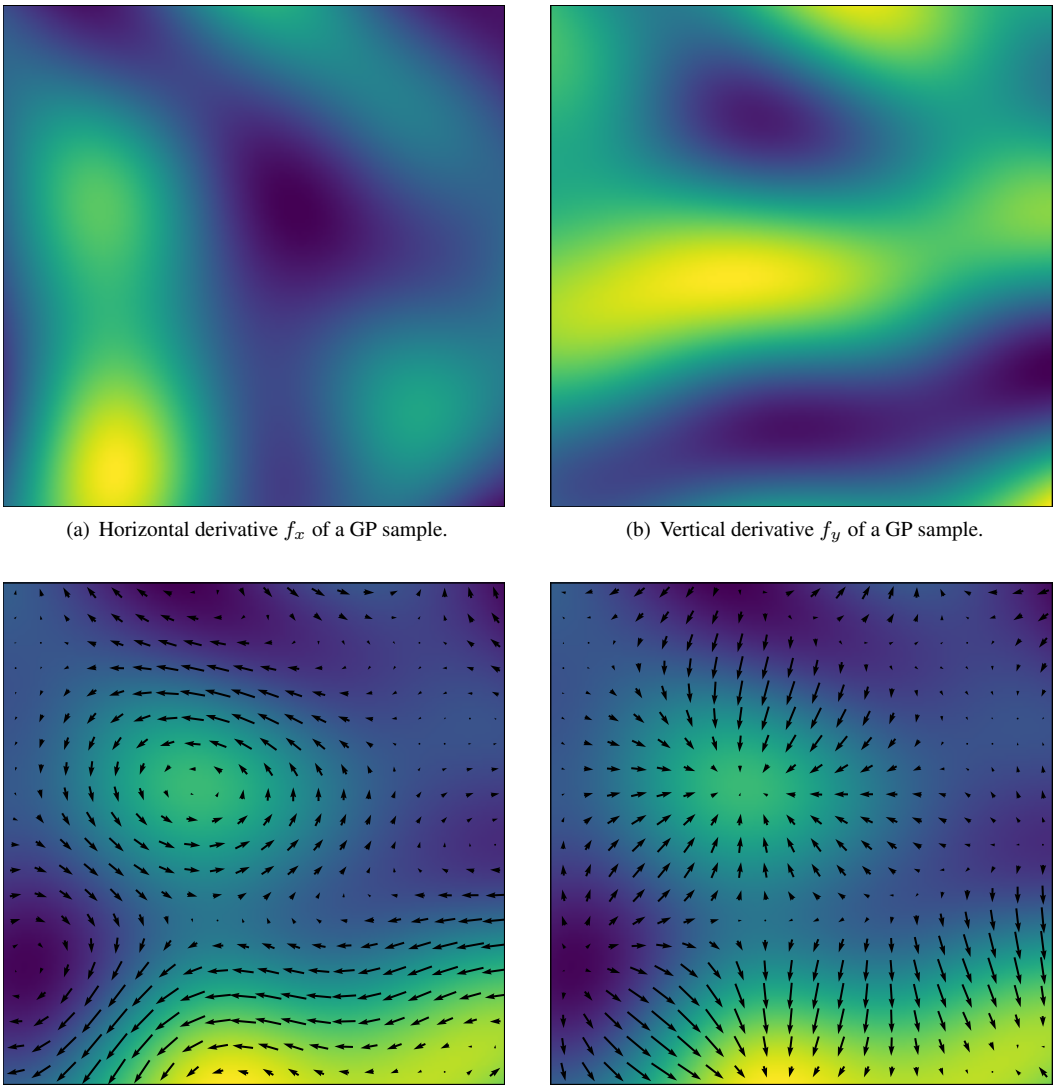
$$\boldsymbol{X}(\boldsymbol{p}) \coloneqq \left(\frac{\partial \psi_\delta}{\partial y}(\boldsymbol{p}), -\frac{\partial \psi_\delta}{\partial x}(\boldsymbol{p})\right)^\top, \tag{26}$$

as for the divergence-free fields of Div-curl-free. Its net circulation, that is, its circulation around a closed curve that encloses all vortices far from them, is

$$\Gamma \approx \sum_{n=1}^{N_\text{v}} \Gamma_n, \tag{27}$$

with equality when $\delta = 0$.

In our experiments, $N_\text{v} = 3$, and every complex draws new vortex locations, independently and uniformly in the meshed square $[-1, 1]^2$, and new strengths, independently from the standard normal distribution. Each generated dataset consists of 100 attributed SCs (200 for the experiments in Appendix D), obtained by applying the de Rham map to the resulting vector fields on one randomly generated triangular mesh shared by all its complexes (Figure 6); each repetition draws a new mesh. The label is "0" if $\sum_{n=1}^{N_\text{v}} \Gamma_n \geq 0$ and "1" otherwise.



(a) Horizontal derivative f_x of a GP sample.

(b) Vertical derivative f_y of a GP sample.

(c) Divergence-free field $(f_y, -f_x)$ over f.

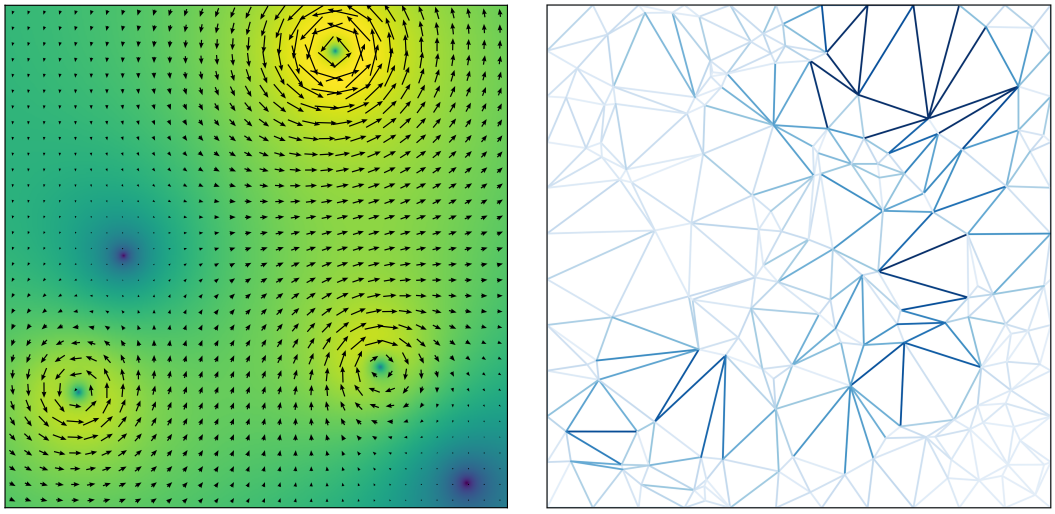
(d) Curl-free field (f_x, f_y) over f.

Figure 5: Illustration of the process that generates the random vector fields of Div-curl-free.

Edge orientation. On both datasets, an edge is oriented from its lower-indexed to its higher-indexed vertex, with the vertices numbered in the order in which they are generated. This order is arbitrary, so the orientation differs from mesh to mesh. Since the edge signal of Vortices changes sign when the orientation is reversed, its sum is comparable only across complexes that share an orientation, which holds here because the complexes of a generated dataset share one mesh. Div-curl-free has a new mesh for every complex, but its label does not depend on the orientation, and the energy is invariant under it.

C.3 Training and evaluation

Overall pipeline. For each SC, we (i) compute the eigendecompositions of its Hodge Laplacians once, split according to the Hodge subspaces; (ii) apply the learnable wavelet filters to the attributes and aggregate the Hodgelet coefficients into the representations $\boldsymbol{r}_{k\bullet}$; and (iii) evaluate the kernel (17) on these representations. The wavelet filter scales and amplitudes, mixture weights (when the mixture aggregation is used), kernel lengthscales and output scales, constant mean, inducing-point locations, variational parameters, and likelihood parameters are optimised jointly by maximising the



(a) Fluid flow with 3 vortices.

(b) Projection onto a triangular mesh.

Figure 6: A data point of Vortices: (a) the flow, with the colour giving its speed (log scale), and (b) its projection onto the mesh of its dataset (150 vertices), with the shade of an edge giving the absolute value of the edge signal.

γ-weighted ELBO with stochastic gradients; only the eigendecompositions are precomputed, and gradients flow from the objective back into the representations. The remaining hyperparameters are fixed for each benchmark (see below) rather than selected within the evaluation protocol. Predictions for a new SC follow from the posterior predictive distribution (3), with the posterior replaced by its variational approximation.

Within each of the five repetitions described at the beginning of this appendix, we run five-fold cross-validation, stratified for classification tasks. For each fold, a model is trained on the four remaining folds and predicts the held-out fold, so that every complex receives exactly one out-of-fold prediction. The metrics (accuracy or mean squared error, NLPD, and ECE) are computed once from these pooled out-of-fold predictions. We report the mean and the standard error over the five repetitions, that is, the sample standard deviation divided by $\sqrt{5}$; the standard error is never computed across folds, which are correlated within a repetition. For TUDataset and MoleculeNet, the repetitions reuse the same graphs with different splits, so their standard errors reflect the variability of the splits and of the training, but not that of the data. For classification, the ECE [Guo et al., 2017] uses ten equal-width bins of the predicted probability of the positive class, except on ieee24 in Table 8, where it uses the probability of the predicted class.

The GPs are trained on the full training folds for a fixed number of epochs, without early stopping. The neural baselines hold out $1/9$ of the training folds as a validation split for early stopping (patience of 50 epochs).

C.4 Model configurations

In Tables 4 and 5, we record the hyperparameters used for the GNN baselines on each dataset. All GNNs use the Adam optimiser and a batch size of 32, map the vertex attributes to the hidden dimension with a linear layer, apply the message-passing layers with ReLU activations and dropout, pool the vertex embeddings with a global sum or mean readout, and predict with a two-layer MLP. We use the PyTorch Geometric layers: GIN uses the edge attributes as in Hu et al. [2020] (GINE), GAT and Transformer use them in their attention mechanism, with four heads, and GCN uses the vertex attributes only; all other layer settings are the PyTorch Geometric defaults. Regression targets are standardised with the statistics of the training folds, excluding the validation split. The baselines of the scaling, Div-curl-free calibration, MANTRA, and further-baseline experiments of Appendix D are configured as in Table 6.

All our Gaussian process models use the additive kernel (17) (for WTGP, with one term per simplex dimension), whose component kernels $\kappa_{k\bullet}$ are radial basis function (RBF) kernels with one lengthscale per feature. The GPs have a constant mean, and regression targets are standardised with the statistics of the training folds. Likewise, each entry of the Hodgelet representations is standardised with its mean and standard deviation over the training fold, computed once at initialisation; this keeps the RBF kernels well conditioned, since the energy and the sum of the Hodgelet coefficients add one term per simplex and therefore grow with the size of the complex, but does not make the representations invariant to size. The inducing points are initialised by k-means on the standardised training representations. The likelihood is Bernoulli with a probit link for binary tasks, a softmax over one latent GP per class for multiclass tasks, and Gaussian for regression.

For the Hodgelet representations, we use, for all k, the scaling function $a_k(x) = e^{-x}$, the wavelet function $b_k(x) = x\,e^{1-x}$, which peaks at $b_k(1) = 1$, and $F_k = 3$ wavelet filters with $J_k = 3$ bandpass scales each; the scales α_k and β_{kl} of the wavelet filters (Section 4.1) are initialised uniformly in $[4, 6]$ and $[0.1, 5]$, respectively, and the filter amplitudes η_k and η_{kl} at 1.

We use sparse variational GPs with $M = 400$ inducing points on FreeSolv and ESOL; on MUTAG and ieee24, M is the size of a training fold, up to 300: 150 or 151 on MUTAG, 296 to 300 on ieee24-binary, 276 to 286 on ieee24-multiclass, and 96 to 100 on ieee24-regression. On MUTAG and ieee24-binary, the accuracy still improves up to these values (see Table 16). We use $M = 40$ on the vector-field datasets and MANTRA, and on AIDS, where all models are close to perfect accuracy. Unless stated otherwise, all GPs are trained for 200 epochs (400 on FreeSolv and 700 on ieee24-regression) with batch size 32.

Lengthscale initialisation. The lengthscales start at the GPyTorch default of $\ln 2$, except on FreeSolv and ESOL. On these two molecular regression benchmarks, this default leaves the vertex kernels almost diagonal at initialisation. We therefore initialise all lengthscales of each component kernel to the median pairwise Euclidean distance between the standardised representations of the training complexes for that component, computed once per training fold at the initial wavelet parameters [the median heuristic; Garreau et al., 2017]. The lengthscales are then optimised as usual.

Aggregation functions. The aggregation function is fixed per benchmark. We consider the energy, the sum, the mixture, and, for attributes that are not cochains, the orientation-averaged sum and energy (15). For the Hodgelet coefficients $\hat{\boldsymbol{x}}_{k\bullet} \in \mathbb{R}^{N_k}$, $\bullet \in \{e, c, h\}$, the energy is $\|\hat{\boldsymbol{x}}_{k\bullet}\|^2 = \sum_{i=1}^{N_k} (\hat{\boldsymbol{x}}_{k\bullet})_i^2$, the sum is $\sum_{i=1}^{N_k} (\hat{\boldsymbol{x}}_{k\bullet})_i$, and the mixture is the learnable weighted combination

$$\omega_1 \sum_{i=1}^{N_k} (\hat{\boldsymbol{x}}_{k\bullet})_i + \omega_2 \|\hat{\boldsymbol{x}}_{k\bullet}\|^2 + \omega_3 \max_i (\hat{\boldsymbol{x}}_{k\bullet})_i + \omega_4 \min_i (\hat{\boldsymbol{x}}_{k\bullet})_i,$$

with mixture weights $\omega_1, \ldots, \omega_4 \in \mathbb{R}$, initialised at $1/4$. The wavelet coefficients of WTGP are aggregated in the same way. On the vector fields, we use the energy for Div-curl-free and the sum for Vortices, whose label is the sign of the net circulation; the energy does not contain this sign and performs poorly on Vortices (see Table 10). For TUDataset and MoleculeNet, we use the mixture for vertex attributes and the orientation-averaged sum and energy (15) for the bond attributes, which are not cochains. For ieee24, all graphs share the grid and hence one orientation, which allows us to use the mixture throughout. For MANTRA, we use the mixture aggregation on the triangle features. The ablations in Appendix D use the aggregation of the corresponding benchmark.

Optimisation. To optimise the ELBO, we introduce a scaling constant γ that balances the expected log-likelihood term against the Kullback–Leibler (KL) regularisation term. Dividing the objective by γ shows that this corresponds to variational inference with the tempered likelihood $p(\boldsymbol{y} \mid f)^{1/\gamma}$; when $\gamma < 1$, this reduces the regularising effect of the prior. We use $\gamma = 0.1$ for all our GP models. For training, we use the AdamW optimiser (AMSGrad variant), with learning rate 0.01 for the wavelet, mixture, kernel, mean, and inducing-point parameters, 0.001 for the variational and likelihood parameters, weight decay 10^{-3}, and gradient clipping at norm 1.

C.5 Compute details

All reported results were computed on a single Apple M3 machine (8 CPU cores, 16 GB of memory), on CPU, in double precision, except for the graph and simplicial neural baselines, which use

Table 4: Configurations used for the graph neural network baselines in Table 1.

	MUTAG	AIDS	FreeSolv	ESOL
Hidden dim.	128	128	64	32
Num. layers	5	5	3	5
Dropout	0.5	0.5	0	0
Weight decay	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
Readout	Sum	Sum	Mean	Sum
Learning rate	0.01	0.01	0.01	0.01
Max. epochs	200	200	300	200
Task	Binary	Binary	Regression	Regression

Table 5: Configurations used for the graph neural network baselines in Table 2.

	ieee24-binary	ieee24-multiclass	ieee24-regression
Hidden dim.	32	32	32
Num. layers	3	3	3
Dropout	0	0	0
Weight decay	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
Readout	Sum	Sum	Mean
Learning rate	0.001	0.001	0.001
Max. epochs	200	200	200
Task	Binary	Multiclass	Regression

single precision. A full five-repetition evaluation of one GP configuration takes a few minutes (MUTAG, and the vector-field datasets of Figure 3 at 100 vertices), up to about twenty minutes (ieee24-regression, ieee24-binary), up to an hour (ieee24-multiclass), and up to about two and a half hours (AIDS, FreeSolv, and ESOL), with several evaluations running in parallel; see Table 17 for timings.

D Additional experiments

All accuracy and calibration results in this appendix follow the evaluation protocol of Appendix C.3: five independent repetitions, five-fold cross-validation with pooled out-of-fold predictions, and the mean and standard error over the repetitions. Unless stated otherwise, the vector-field experiments use 200 complexes on meshes with 100 vertices, and ieee24-binary uses about 376 graphs, with the same configuration as in Table 2.

D.1 Calibration of the main benchmarks

Tables 7, 8, and 9 report the NLPD and ECE of the models in Tables 1 and 2, computed from the same pooled out-of-fold predictions. For classification, the NLPD and ECE of the neural baselines are computed from their softmax probabilities. For regression, they only output point predictions, so we use a Gaussian predictive distribution centred on the point prediction, with variance equal to the mean squared error on the validation split used for early stopping in that fold. For all models, the ECE for regression, a measure of quantile calibration [Kuleshov et al., 2018], is the mean, over $p \in \{0.05, 0.15, \ldots, 0.95\}$, of the absolute difference between p and the fraction of targets below their predictive p-quantile.

On the classification benchmarks, HTGP (hybrid) or HTGP (edge) attains the lowest NLPD and ECE on MUTAG and on both ieee24 tasks; on ieee24-multiclass, WTGP (edge) has the same NLPD as HTGP (edge). For example, on MUTAG, the NLPD of HTGP (hybrid) is 0.296, compared with 0.452 to 0.503 for the GNNs. On AIDS, all models have an NLPD below 0.033 and an ECE below 0.007. On the regression benchmarks, the NLPD of the GP models largely follows the ranking of their mean squared errors, and the GP models that use the vertex attributes attain a lower NLPD than all neural baselines. The ECE does not follow this ranking: the edge-only GPs attain the lowest ECE on FreeSolv, and HTGP (edge) on ESOL, despite their large errors, and the neural baselines, whose

Table 6: Configurations of the Transformer in the scaling, Div-curl-free calibration, and MANTRA experiments, and of the further baselines of Tables 12 and 18. The neural baselines hold out $1/9$ of the training fold for early stopping (patience of 50 epochs) and use single precision.

Method	Model	Training
Transformer	Architecture described above, hidden dimension 32, three layers, no dropout, mean readout	Adam, learning rate 10^{-3}, weight decay $5 \cdot 10^{-5}$, batch size 32, at most 200 epochs
SNN	Two layers of width 32 with ReLU, polynomial filters of order 3 in $\boldsymbol{L}_1$ ($\boldsymbol{L}_2$ on MANTRA), scaled by its largest eigenvalue, sum readout, linear head	Adam, learning rate 10^{-2}, weight decay 10^{-4}, full batch, at most 200 epochs
SCNN	As SNN, with an identity term and separate filters of order 2 in $\boldsymbol{L}_1^{\text{low}}$ and $\boldsymbol{L}_1^{\text{up}}$ (on MANTRA, in $\boldsymbol{L}_2^{\text{low}} = \boldsymbol{L}_2$ only, since $\boldsymbol{L}_2^{\text{up}} = \boldsymbol{0}$), each scaled by its largest eigenvalue	As SNN
WL-SVM	Weisfeiler–Lehman subtree features (3 iterations, label counts normalised to unit length), linear kernel, SVM with $C = 1$ and Platt-scaled probabilities	–
WL-GP	Features of WL-SVM, scaled linear kernel, constant mean, Bernoulli likelihood, $M = 40$ inducing points initialised by k-means, KL term not down-weighted ($\gamma = 1$)	Adam, learning rate 10^{-2}, 300 full-batch epochs, no early stopping

variance is fitted on the validation split, attain a lower ECE than the GP models on ieee24-regression (0.041 to 0.060 versus 0.071 to 0.093).

Table 7: NLPD (↓) and ECE (↓) of the models in Table 1 on MUTAG and AIDS. Colours indicate the best, second-best, and third-best result for each column.

	MUTAG		AIDS	
	NLPD	ECE	NLPD	ECE
GCN (vertex)	0.467 ± 0.020	0.102 ± 0.013	$\mathbf{0.0229} \pm 0.0009$	0.0049 ± 0.0008
GAT (hybrid)	0.503 ± 0.021	0.091 ± 0.016	$\mathbf{0.0226} \pm 0.0007$	0.0049 ± 0.0008
GIN (hybrid)	0.452 ± 0.023	0.099 ± 0.016	$\mathbf{0.0235} \pm 0.0009$	0.0057 ± 0.0010
Transformer (hybrid)	0.490 ± 0.037	0.086 ± 0.008	$\mathbf{0.0235} \pm 0.0007$	0.0048 ± 0.0006
WTGP (hybrid)	0.338 ± 0.015	0.067 ± 0.006	0.0240 ± 0.0011	$\mathbf{0.0031} \pm 0.0005$
WTGP (vertex)	0.424 ± 0.008	$\mathbf{0.066} \pm 0.011$	0.0318 ± 0.0009	0.0051 ± 0.0004
WTGP (edge)	$\mathbf{0.315} \pm 0.014$	0.077 ± 0.007	0.0281 ± 0.0010	$\mathbf{0.0030} \pm 0.0002$
HTGP (hybrid)	$\mathbf{0.296} \pm 0.011$	$\mathbf{0.064} \pm 0.004$	0.0264 ± 0.0011	$\mathbf{0.0033} \pm 0.0003$
HTGP (vertex)	0.449 ± 0.018	0.067 ± 0.010	0.0288 ± 0.0011	0.0050 ± 0.0004
HTGP (edge)	$\mathbf{0.297} \pm 0.009$	$\mathbf{0.055} \pm 0.002$	0.0273 ± 0.0011	0.0034 ± 0.0003

D.2 Aggregation function

Table 10 compares the energy, sum, and mixture aggregations for the edge models.

The best aggregation depends on the task. On Div-curl-free, the energy performs best, closely followed by the mixture, whereas the sum performs poorly (0.599 for HTGP), since it depends on the signs of the edge values and hence on the edge orientation, which differs from complex to complex. On Vortices, the label is the sign of the circulation, which the energy discards: the models with the energy are at chance level, whereas the sum reaches 0.814 for HTGP and 0.767 for WTGP. On Vortices, the mixture lies between the energy and the sum. On ieee24-binary, the

Table 8: NLPD (↓) and ECE (↓) of the models in Table 2 on ieee24-binary and ieee24-multiclass. Colours indicate the best, second-best, and third-best result for each column.

	ieee24-binary		ieee24-multiclass	
	NLPD	ECE	NLPD	ECE
GCN (vertex)	0.513 ± 0.012	0.038 ± 0.004	0.764 ± 0.043	0.054 ± 0.010
GAT (hybrid)	0.329 ± 0.015	$\mathbf{0.031} \pm 0.004$	0.397 ± 0.020	0.033 ± 0.004
GIN (hybrid)	0.308 ± 0.019	0.033 ± 0.005	0.348 ± 0.014	$\mathbf{0.028} \pm 0.003$
Transformer (hybrid)	0.306 ± 0.030	0.037 ± 0.007	0.356 ± 0.014	0.036 ± 0.002
WTGP (hybrid)	$\mathbf{0.284} \pm 0.021$	$\mathbf{0.024} \pm 0.007$	$\mathbf{0.267} \pm 0.008$	$\mathbf{0.029} \pm 0.003$
WTGP (vertex)	0.445 ± 0.013	0.038 ± 0.008	0.632 ± 0.027	0.050 ± 0.004
WTGP (edge)	0.339 ± 0.041	0.035 ± 0.008	$\mathbf{0.235} \pm 0.011$	0.030 ± 0.004
HTGP (hybrid)	$\mathbf{0.257} \pm 0.018$	$\mathbf{0.031} \pm 0.003$	$\mathbf{0.277} \pm 0.014$	0.032 ± 0.002
HTGP (vertex)	0.437 ± 0.012	0.038 ± 0.006	0.630 ± 0.040	0.039 ± 0.007
HTGP (edge)	$\mathbf{0.246} \pm 0.012$	$\mathbf{0.021} \pm 0.003$	$\mathbf{0.235} \pm 0.007$	$\mathbf{0.027} \pm 0.003$

Table 9: NLPD (↓) and ECE (↓) of the models in Tables 1 and 2 on the regression benchmarks. Colours indicate the best, second-best, and third-best result for each column.

	FreeSolv		ESOL		ieee24-regression	
	NLPD	ECE	NLPD	ECE	NLPD	ECE
GCN (vertex)	1.920 ± 0.066	$\mathbf{0.030} \pm 0.004$	1.058 ± 0.007	$\mathbf{0.015} \pm 0.001$	-2.266 ± 0.078	$\mathbf{0.047} \pm 0.001$
GAT (hybrid)	2.047 ± 0.053	0.039 ± 0.005	1.057 ± 0.015	0.029 ± 0.006	-2.294 ± 0.082	$\mathbf{0.041} \pm 0.003$
GIN (hybrid)	1.898 ± 0.046	0.041 ± 0.006	1.393 ± 0.048	$\mathbf{0.015} \pm 0.003$	-2.398 ± 0.045	0.060 ± 0.011
Transformer (hybrid)	1.915 ± 0.079	0.042 ± 0.004	1.086 ± 0.023	$\mathbf{0.018} \pm 0.002$	-2.380 ± 0.042	$\mathbf{0.058} \pm 0.008$
WTGP (hybrid)	$\mathbf{1.318} \pm 0.023$	0.045 ± 0.002	$\mathbf{0.939} \pm 0.010$	0.025 ± 0.004	$\mathbf{-2.858} \pm 0.027$	0.079 ± 0.003
WTGP (vertex)	$\mathbf{1.339} \pm 0.024$	0.042 ± 0.002	$\mathbf{0.941} \pm 0.007$	0.027 ± 0.004	-2.649 ± 0.047	0.071 ± 0.004
WTGP (edge)	2.554 ± 0.016	$\mathbf{0.019} \pm 0.002$	1.641 ± 0.007	$\mathbf{0.015} \pm 0.002$	-2.533 ± 0.102	0.082 ± 0.004
HTGP (hybrid)	$\mathbf{1.415} \pm 0.008$	$\mathbf{0.032} \pm 0.001$	$\mathbf{0.972} \pm 0.006$	0.023 ± 0.002	$\mathbf{-2.996} \pm 0.051$	0.088 ± 0.009
HTGP (vertex)	1.445 ± 0.008	0.038 ± 0.004	0.991 ± 0.012	0.028 ± 0.005	-2.645 ± 0.032	0.077 ± 0.005
HTGP (edge)	2.533 ± 0.008	$\mathbf{0.019} \pm 0.001$	1.641 ± 0.006	$\mathbf{0.014} \pm 0.004$	$\mathbf{-2.927} \pm 0.059$	0.093 ± 0.007

three aggregations perform similarly (between 0.900 and 0.915), except for WTGP (edge) with the mixture (0.843).

Table 10: Accuracy for different aggregation functions. The best result in each row is in bold.

Dataset	Model	Energy	Sum	Mixture
Div-curl-free	HTGP (edge)	$\mathbf{0.898} \pm 0.005$	0.599 ± 0.030	0.888 ± 0.005
Div-curl-free	WTGP (edge)	$\mathbf{0.823} \pm 0.019$	0.551 ± 0.020	0.816 ± 0.021
Vortices	HTGP (edge)	0.534 ± 0.026	$\mathbf{0.814} \pm 0.023$	0.731 ± 0.056
Vortices	WTGP (edge)	0.519 ± 0.031	$\mathbf{0.767} \pm 0.036$	0.754 ± 0.049
ieee24-binary	HTGP (edge)	$\mathbf{0.915} \pm 0.008$	0.902 ± 0.006	0.902 ± 0.007
ieee24-binary	WTGP (edge)	0.900 ± 0.009	$\mathbf{0.904} \pm 0.002$	0.843 ± 0.025

D.3 Hodge components

Table 11 reports the accuracy of HTGP (edge) when only one Hodge component is used, compared with all components. The meshes have no holes, so their harmonic subspace is empty.

The benchmarks rely on different subspaces. On Div-curl-free, the exact and co-exact components alone reach 0.769 and 0.860, and combining them improves the accuracy to 0.898; both carry information, since the label compares the divergence-free and curl-free parts of the field. On Vortices, the exact and co-exact components alone reach 0.728 and 0.740, and combining them improves the accuracy to 0.814. On ieee24-binary, the exact component carries most of the information (0.892, compared with 0.902 for all components), followed by the harmonic component (0.850); the co-

exact component, which is supported on the few triangles of the power grids, is the least informative (0.758).

Table 11: Ablation of the Hodge components (HTGP (edge); energy on Div-curl-free, sum on Vortices, mixture on ieee24-binary). "–" indicates an empty subspace. The best result in each row is in bold.

Dataset	Harmonic	Co-exact	Exact	All
Div-curl-free	–	0.860 ± 0.014	0.769 ± 0.006	$\mathbf{0.898} \pm 0.005$
Vortices	–	0.740 ± 0.052	0.728 ± 0.016	$\mathbf{0.814} \pm 0.023$
ieee24-binary	0.850 ± 0.010	0.758 ± 0.011	0.892 ± 0.010	$\mathbf{0.902} \pm 0.007$

D.4 Comparison with further baselines

Table 12 compares the GP models with four further baselines: the simplicial neural network (SNN) of Ebli et al. [2020] and the simplicial convolutional neural network (SCNN) of Yang et al. [2022], both with a sum readout, and the Weisfeiler–Lehman subtree kernel [Shervashidze et al., 2011] combined with an SVM (WL-SVM) or a GP classifier (WL-GP). The simplicial networks use the same edge signals and Hodge Laplacians as our models; on MUTAG, they therefore depend on the atom numbering through the bond attributes. The WL kernels need a discrete label on each vertex. On the vector-field datasets, we compute two numbers at each vertex, the divergence $(-\boldsymbol{B}_1\boldsymbol{x}_1)_v$ and the incident energy $(|\boldsymbol{B}_1|\boldsymbol{x}_1^2)_v$ of the edge signal, split the range of each into four intervals at its quartiles over all vertices of the dataset, and label the vertex by the pair of intervals into which its two numbers fall, which gives 16 possible labels. On MUTAG, the labels are the atom types. Table 6 gives the configurations of all four baselines.

HTGP attains the highest accuracy and the lowest NLPD on all three datasets. On Div-curl-free, all four baselines are close to chance level, whereas HTGP and WTGP reach 0.898 and 0.823. On Vortices, the simplicial networks reach 0.632 and 0.645, compared with 0.814 for HTGP, and the WL kernels are close to chance level, since their vertex labels do not encode the sign of the circulation. On MUTAG, the simplicial networks also fall behind, whereas the WL kernels perform reasonably.

Table 12: Comparison with further baselines. The vector-field datasets use the edge signals, with the energy on Div-curl-free and the sum on Vortices for the GP models; on MUTAG, the GP models use vertex and edge attributes. Colours indicate the best, second-best, and third-best result for each dataset and metric.

Dataset	Method	NLPD (↓)	ECE (↓)	Accuracy (↑)
Div-curl-free	HTGP (edge)	$\mathbf{0.245} \pm 0.008$	$\mathbf{0.041} \pm 0.006$	$\mathbf{0.898} \pm 0.005$
Div-curl-free	WTGP (edge)	$\mathbf{0.399} \pm 0.038$	0.053 ± 0.005	$\mathbf{0.823} \pm 0.019$
Div-curl-free	SNN	0.769 ± 0.025	0.145 ± 0.016	$\mathbf{0.516} \pm 0.013$
Div-curl-free	SCNN	0.727 ± 0.006	0.117 ± 0.009	0.509 ± 0.011
Div-curl-free	WL-SVM	0.696 ± 0.002	$\mathbf{0.045} \pm 0.013$	0.481 ± 0.013
Div-curl-free	WL-GP	$\mathbf{0.693} \pm 0.001$	$\mathbf{0.025} \pm 0.010$	0.500 ± 0.016
Vortices	HTGP (edge)	$\mathbf{0.434} \pm 0.043$	$\mathbf{0.058} \pm 0.009$	$\mathbf{0.814} \pm 0.023$
Vortices	WTGP (edge)	$\mathbf{0.510} \pm 0.055$	0.071 ± 0.011	$\mathbf{0.767} \pm 0.036$
Vortices	SNN	0.652 ± 0.017	0.105 ± 0.013	0.632 ± 0.018
Vortices	SCNN	$\mathbf{0.618} \pm 0.024$	0.090 ± 0.014	$\mathbf{0.645} \pm 0.042$
Vortices	WL-SVM	0.689 ± 0.004	$\mathbf{0.046} \pm 0.008$	0.542 ± 0.017
Vortices	WL-GP	0.680 ± 0.005	$\mathbf{0.036} \pm 0.011$	0.555 ± 0.009
MUTAG	HTGP (hybrid)	$\mathbf{0.296} \pm 0.011$	$\mathbf{0.064} \pm 0.004$	$\mathbf{0.878} \pm 0.009$
MUTAG	WTGP (hybrid)	$\mathbf{0.338} \pm 0.015$	$\mathbf{0.067} \pm 0.006$	$\mathbf{0.863} \pm 0.010$
MUTAG	SNN	0.526 ± 0.031	0.121 ± 0.007	0.732 ± 0.017
MUTAG	SCNN	0.532 ± 0.049	0.118 ± 0.010	0.738 ± 0.016
MUTAG	WL-SVM	0.429 ± 0.002	0.136 ± 0.008	0.794 ± 0.003
MUTAG	WL-GP	$\mathbf{0.374} \pm 0.005$	$\mathbf{0.114} \pm 0.009$	$\mathbf{0.802} \pm 0.006$

D.5 Effect of the 2-simplices

On the meshes, the 2-simplices are native; on the graph benchmarks, they are added by filling every 3-clique. The number of triangles varies widely: the meshes have 226 triangles per complex, ieee24 about one, and MUTAG none. Table 13 compares HTGP (edge) with and without the 2-simplices; without them, the co-exact subspace vanishes.

Removing the 2-simplices has little effect on ieee24-binary (from 0.902 to 0.899), but lowers the accuracy on Div-curl-free from 0.898 to 0.838 and on Vortices from 0.814 to 0.754. Hence, on ieee24-binary, separating the exact component from the rest of the edge signal already gives most of the benefit of the Hodge decomposition, whereas on the meshes, the co-exact subspace created by the 2-simplices matters as well.

Table 13: Accuracy of HTGP (edge) with and without 2-simplices (energy on Div-curl-free, sum on Vortices, mixture on ieee24-binary). The best result in each row is in bold.

Dataset	HTGP (edge) with 2-simplices	HTGP (edge) without 2-simplices
Div-curl-free	$\mathbf{0.898} \pm 0.005$	0.838 ± 0.009
Vortices	$\mathbf{0.814} \pm 0.023$	0.754 ± 0.020
ieee24-binary	$\mathbf{0.902} \pm 0.007$	0.899 ± 0.009

D.6 Data efficiency on ieee24

Table 14 compares HTGP (hybrid) with the Transformer on ieee24-binary as the number of graphs N grows. HTGP uses the energy aggregation, 600 training epochs, and $M = 80$ inducing points, capped at the size of a training fold (75 to 77 graphs) at $N = 100$. For each N, we draw $N/2$ graphs per class and keep only the connected graphs. HTGP is more accurate up to $N = 2000$; the two models are tied at $N = 2500$, and the Transformer is slightly more accurate at $N = 3000$ (0.951 versus 0.947). HTGP attains a lower NLPD at every size, including when the Transformer matches or exceeds its accuracy, and a lower ECE up to $N = 2500$; at $N = 3000$, the two ECEs are almost equal (0.017 versus 0.016).

Table 14: Scaling with the number of graphs N on ieee24-binary: HTGP (hybrid) versus the Transformer. Accuracy (↑), NLPD (↓), and ECE (↓); standard errors are given for the accuracy.

N	HTGP acc.	Transf. acc.	HTGP NLPD	Transf. NLPD	HTGP ECE	Transf. ECE
100	0.855 ± 0.018	0.790 ± 0.030	0.461	0.499	0.106	0.110
500	0.917 ± 0.008	0.891 ± 0.004	0.270	0.290	0.037	0.045
1000	0.927 ± 0.004	0.912 ± 0.004	0.205	0.247	0.024	0.029
1500	0.934 ± 0.004	0.924 ± 0.006	0.184	0.217	0.017	0.022
2000	0.943 ± 0.001	0.930 ± 0.002	0.159	0.201	0.019	0.021
2500	0.945 ± 0.003	0.945 ± 0.004	0.149	0.168	0.013	0.018
3000	0.947 ± 0.003	0.951 ± 0.004	0.143	0.156	0.017	0.016

D.7 Calibration on Div-curl-free

Table 15 reports the NLPD, ECE, and accuracy on Div-curl-free as the number of complexes N grows (energy aggregation). The Transformer stays close to chance level at every dataset size (accuracy 0.494 to 0.528); its ECE is low only because its predicted probabilities stay close to $1/2$, as its NLPD of about 0.69, that of always predicting $1/2$, shows. HTGP attains the lowest NLPD and the highest accuracy throughout, and a lower ECE than WTGP at every dataset size.

D.8 Number of inducing points

Table 16 reports the accuracy of HTGP (hybrid) as a function of the number of inducing points M, on MUTAG and ieee24-binary with the configurations of Tables 1 and 2. On both datasets, the accuracy increases quickly with M and is within two points of its maximum from $M = 50$ on. It is highest with the M of Tables 1 and 2 (150 or 151 on MUTAG, 296 to 300 on ieee24-binary):

Table 15: Calibration on Div-curl-free versus the number of complexes N. For each N, the best result for each metric is in bold.

N	Model	NLPD (↓)	ECE (↓)	Accuracy (↑)
50	HTGP (edge)	$\mathbf{0.305} \pm 0.036$	0.114 ± 0.010	$\mathbf{0.880} \pm 0.030$
	WTGP (edge)	0.499 ± 0.063	0.121 ± 0.007	0.776 ± 0.047
	Transformer	0.692 ± 0.003	$\mathbf{0.061} \pm 0.017$	0.520 ± 0.021
100	HTGP (edge)	$\mathbf{0.271} \pm 0.006$	0.057 ± 0.005	$\mathbf{0.888} \pm 0.010$
	WTGP (edge)	0.423 ± 0.039	0.092 ± 0.012	0.786 ± 0.028
	Transformer	0.705 ± 0.011	$\mathbf{0.027} \pm 0.009$	0.528 ± 0.008
250	HTGP (edge)	$\mathbf{0.232} \pm 0.012$	0.040 ± 0.002	$\mathbf{0.906} \pm 0.006$
	WTGP (edge)	0.397 ± 0.022	0.049 ± 0.004	0.820 ± 0.017
	Transformer	0.697 ± 0.002	$\mathbf{0.037} \pm 0.010$	0.494 ± 0.008
500	HTGP (edge)	$\mathbf{0.226} \pm 0.007$	0.029 ± 0.001	$\mathbf{0.912} \pm 0.007$
	WTGP (edge)	0.410 ± 0.011	0.039 ± 0.003	0.810 ± 0.009
	Transformer	0.693 ± 0.000	$\mathbf{0.011} \pm 0.001$	0.503 ± 0.002
1000	HTGP (edge)	$\mathbf{0.213} \pm 0.004$	0.021 ± 0.002	$\mathbf{0.915} \pm 0.004$
	WTGP (edge)	0.400 ± 0.003	0.026 ± 0.003	0.815 ± 0.003
	Transformer	0.692 ± 0.001	$\mathbf{0.009} \pm 0.004$	0.515 ± 0.009

HTGP (hybrid) then reaches 0.878 on MUTAG and 0.906 on ieee24-binary, compared with 0.865 at $M = 100$ and 0.898 at $M = 200$, respectively.

Table 16: Effect of the number of inducing points M on the accuracy of HTGP (hybrid). On MUTAG, M is capped at the size of a training fold (150 or 151 graphs); larger values are not shown.

M	MUTAG	ieee24-binary
1	0.665 ± 0.000	0.605 ± 0.020
5	0.810 ± 0.003	0.850 ± 0.009
10	0.831 ± 0.004	0.862 ± 0.009
50	0.870 ± 0.003	0.886 ± 0.008
100	0.865 ± 0.009	0.894 ± 0.007
200	–	0.898 ± 0.007

D.9 Training time

Table 17 compares the training time of HTGP and the Transformer over 200 epochs. HTGP is cheaper per epoch on the meshes of Div-curl-free, but not on the small ieee24 graphs. The eigendecomposition of the Hodge Laplacians is performed once as preprocessing and takes at most 3.6 s here. In terms of total time, HTGP is faster than the Transformer on Div-curl-free and about 2.5 times slower on ieee24-binary, and the sparse variational approximation keeps its cost linear in the number of complexes.

Table 17: Training time of HTGP and the Transformer for 200 epochs on one training fold (mean of three runs on one CPU thread, with no other experiments running), with the configurations of Tables 14 and 15: HTGP (hybrid) on ieee24-binary and HTGP (edge) on Div-curl-free. The total of HTGP includes the eigendecomposition for all complexes and the initialisation.

Dataset	HTGP ms/epoch	Transf. ms/epoch	Eigendecomposition	Total (HTGP / Transf.)
ieee24-binary ($N = 300$, $M = 80$)	67.5	27.3	0.08 s	13.6 s / 5.5 s
ieee24-binary ($N = 1000$, $M = 80$)	209.8	86.6	0.27 s	42.4 s / 17.3 s
Div-curl-free ($N = 200$, $M = 40$)	77.3	128.8	3.58 s	19.2 s / 25.8 s

D.10 Topological prediction on MANTRA

To test our models on data with native higher-order structure, we consider MANTRA [Ballester et al., 2025], a dataset of triangulations of closed surfaces, and predict whether a surface is orientable. Orientability is determined by how the triangles are glued together: for a connected closed surface, the harmonic subspace of the triangles, $\ker(\boldsymbol{L}_2)$, has dimension 1 if the surface is orientable and 0 otherwise. In each repetition, we draw a balanced sample of 500 surfaces (250 per class). Since MANTRA has no attributes, we give each triangle eight random numbers, drawn independently from the standard normal distribution and redrawn in each repetition (Appendix C.1). These features carry no information about the label, but their projection onto the harmonic subspace is non-zero with probability one if the surface is orientable, and zero otherwise. HTGP (triangle) and WTGP (triangle) use only the triangle features: HTGP splits them into their exact and harmonic components (the co-exact subspace of the triangles is empty, as there are no 3-simplices), whereas WTGP applies its wavelet filters to all eigenvalues of $\boldsymbol{L}_2$ at once, without separating the components. SNN and SCNN take the same triangle features as input, with polynomial filters in $\boldsymbol{L}_2$ instead of $\boldsymbol{L}_1$. The Transformer operates on the 1-skeleton graph, with the vertex degrees and the numbers of triangles per edge as attributes. The graph carries only partial information about orientability: the Transformer reaches an accuracy of 0.744. SNN and SCNN, whose low-order polynomial filters presumably cannot isolate the harmonic component, stay close to chance level (0.508 and 0.516), whereas HTGP (triangle) and WTGP (triangle) are almost perfect (1.000 and 0.992; Table 18).

Table 18: MANTRA orientability (triangulated surfaces). The best result for each metric is in bold.

Method	NLPD (↓)	ECE (↓)	Accuracy (↑)
HTGP (triangle)	$\mathbf{0.004} \pm 0.000$	$\mathbf{0.003} \pm 0.000$	$\mathbf{1.000} \pm 0.000$
WTGP (triangle)	0.028 ± 0.004	0.007 ± 0.001	0.992 ± 0.001
SNN	0.711 ± 0.006	0.072 ± 0.004	0.508 ± 0.011
SCNN	0.707 ± 0.005	0.069 ± 0.013	0.516 ± 0.003
Transformer	0.544 ± 0.020	0.073 ± 0.014	0.744 ± 0.018