Gaussian process regression with Sliced Wasserstein Weisfeiler-Lehman graph kernels

Anonymous Author
Anonymous Institution

Abstract

Supervised learning has recently garnered significant attention in the field of computational physics due to its ability to effectively extract complex patterns for tasks like solving partial differential equations, or predicting material properties. Traditionally, such datasets consist of inputs given as meshes with a large number of nodes representing the problem geometry (seen as graphs), and corresponding outputs obtained with a numerical solver. This means the supervised learning model must be able to handle large and sparse graphs with continuous attributes. In this work, we focus on Gaussian process regression, for which we introduce the Sliced Wasserstein Weisfeiler-Lehman (SWWL) graph kernel. In contrast to existing graph kernels, the proposed SWWL kernel enjoys positive definiteness and a drastic complexity reduction, which makes it possible to process datasets that were previously impossible to handle. The new kernel is first validated on graph classification for molecular datasets, where the input graphs have a few tens of nodes. The efficiency of the SWWL kernel is then illustrated on graph regression in computational fluid dynamics and solid mechanics, where the input graphs are made up of tens of thousands of nodes.

1 INTRODUCTION

Over the past decade, there has been a growing interest for efficient machine learning algorithms applied to graph data. Graph learning tasks have historically emerged in fields such as biochemistry and social recommendation systems, but very recently, learning physics-based simulations described by partial differential equations (PDEs) has attracted a lot of attention. Indeed, traditional methods such as finite element provide accurate approximations of the PDE solution in computational fluid and solids mechanics, but such mesh-based methods are known for being computationally expensive, and thus difficult to apply to real-world design problems. To circumvent this issue, many efforts have been dedicated to the development of neural networks in order to emulate solutions to physical systems, either by incorporating physical knowledge (Karniadakis et al., 2021), or by designing efficient graph neural networks architectures (Pfaff et al., 2020). In this setting, the supervised learning task involves inputs given as graphs and outputs being physical quantities of interest. We focus here on kernel methods, and more specifically Gaussian process regression, since they are powerful when the sample size is small, such as in computational physics, and when uncertainty quantification is needed. But in most applications the input graphs correspond to finite element meshes with several tens of thousands of nodes and continuous attributes. There is a wealth of literature about kernels between graphs, but few approaches handle such graphs with continuous node attributes, and most of them are intractable with *large* and *sparse* graphs.

Herein, we propose the Sliced Wasserstein Weisfeiler-Lehman (SWWL) kernel as a variant of the Wasserstein Weisfeiler-Lehman (WWL) graph kernel proposed by Togninalli et al. (2019). In a first step, unsupervised node embeddings are built using a continuous Weisfeiler-Lehman scheme. Then, instead of relying on the Wasserstein distance, the sliced Wasserstein distance between the graphs nodes attributes is computed. Figure 1 summarizes the proposed methodology. In addition to the drastic complexity reduction, the sliced Wasserstein (SW) distance is Hilbertian, thus allowing the construction of a positive definite kernel, which is not the case with the Wasserstein

Preliminary work. Under review by AISTATS 2024. Do not distribute.

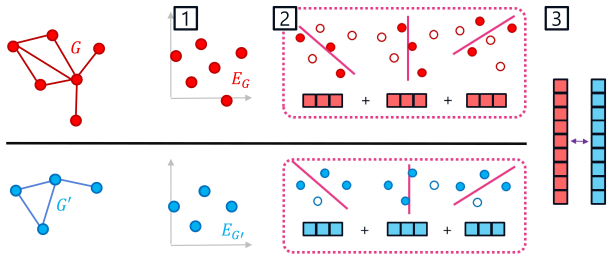

Figure 1: SWWL kernel. Step 1: Graph embedding. Step 2: Projected quantile embeddings with only a few quantiles kept. Step 3: Euclidean distances between embeddings.

distance. Similarly to the WWL kernel, our approach creates unsupervised and non-parametric embeddings. The learning process can thus be broken down into two parts: preprocessing the input graphs, and assembling the Gram matrix with the optimisation of its hyperparameters. This is an advantage over related methods that have features built in a supervised manner. The estimation of SW distances involves random projections and empirical quantiles, and we show experimentally that good accuracies can be obtained with few projections and quantiles, thus resulting in a significant dimension reduction of the graph embedding without degrading performance.

Our contribution is three-fold:

- We design a new graph kernel called the Sliced Wasserstein Weisfeiler-Lehman kernel, which can handle large-scale graphs.
- We show the positive definiteness of this kernel.
- We demonstrate its efficiency when used in Gaussian process regression on high-dimensional simulation datasets from computational physics.

The article is organized as follows. Related work on graph kernels are discussed in Section 2. Section 3 first introduces the problem setting and recalls Gaussian process regression. The proposed SWWL kernel and its properties are described in Section 4. Numerical experiments are then performed in Section 5. The efficiency of the SWWL kernel is assessed first for classification tasks with molecular data, and for regression tasks in the case of mesh-based simulations.

2 RELATED WORK

The problem of assessing similarity between graphs dates back to the 2000s, and a multitude of approaches have been proposed since then (Kriege et al., 2020). Among the rich literature about kernels on graphs (Nikolentzos et al., 2021; Borgwardt et al., 2020), it can be noted that most are not adapted to the case where graphs have continuous node attributes. This seriously hinders their applicability to computational physics, characterized by real-valued physical properties at the mesh nodes.

Complexity of such methods is another limitation: originally designed for molecular datasets where graphs have less than 100 nodes, such kernels do not scale up well to larger graphs. Last but not least, graph kernels often focus on graph structure by matching sub-patterns with the R-convolution framework (Haussler et al., 1999) such as shortest paths, random walks, graphlets or trees. However, this turns out to be irrelevant when the adjacency varies little between several inputs. The approach of Morris et al. (2016) extends some of the previous kernels to the continuous setting using multiple hashing functions.

Recently, many approaches focused on solving the graph-alignment problem using optimal transport and trying to find a matching between well-chosen distributions. The Fused-Gromov distance proposed by Vayer et al. (2018) makes it possible to separate the information provided by the graph structure and its attributes with a trade-off between Wasserstein and Gromov-Wasserstein distances. The recent approach of Kaloga et al. (2022) uses embeddings obtained with Simple Graph Convolutional Network (SGCN) (Wu et al., 2019), a simplified version of GCN reducing the number of trainable parameters. They also replace the Wasserstein distance by the Restricted Projected Wasserstein that uses projections to determine the optimal transport plans and their costs. Other approaches propose to use the Linear Optimal Transport framework by computing transport displacements to a reference template (Kolouri et al. (2020)) or to regularize the Wasserstein distance using both local and global structural information of the graph (Wijesinghe et al. (2021)). However, all approaches using Wasserstein or Gromov-Wasserstein distances suffer from scalability issues and from the impossibility to obtain a positive definite kernel.

3 PRELIMINARIES

We consider the task of learning a function $f : \mathcal{G} \to \mathcal{Y}$ where $\mathcal{Y} = \{0, 1\}$ for classification tasks, and $\mathcal{Y} = \mathbb{R}$ for regression tasks. Here, $\mathcal{G}$ denotes a set of undirected and possibly weighted graphs having continuous attributes. Each graph $G \in \mathcal{G}$ can thus be represented as $G = (V, E, w, \mathbf{F})$ where V is the set of $|V|$ nodes, and E is a set of paired nodes, whose elements are called edges. The d-dimensional attributes of the nodes are gathered in the $(|V| \times d)$ matrix $\mathbf{F} = (\mathbf{F}_u)_{u \in V}$. The edge weights are assigned by the function $w : E \to \mathbb{R}$. The neighborhood of a node $u \in V$ is given by $\mathcal{N}(u) = \{v \in V : \{u, v\} \in E\}$ and its degree is denoted by $\deg(u) = |\mathcal{N}(u)|$.

We assume that we are given a dataset $\mathcal{D}$ consisting of N observations $\mathcal{D} = \{(G_i, y_i)\}_{i=1}^N$, where the input graphs may differ in terms of numbers of nodes and adjacency matrices, *i.e.*, we may have $|V_i| \neq |V_j|$ and/or $E_i \neq E_j$ for some $(i, j) \in \{1, \ldots, N\}^2$. We consider the task of approximating function f with kernel methods and in particular Gaussian process regression.

Gaussian process regression. Gaussian process (GP) regression is a popular Bayesian approach for supervised learning in engineering (Williams and Rasmussen, 2006; Gramacy, 2020; Gu et al., 2018). We assume that we are given noisy training observations $y_i = f(G_i) + \epsilon_i$ for $i \in \{1, \ldots, N\}$ of $f : \mathcal{G} \to \mathbb{R}$ at input locations $\mathbf{G} = (G_i)_{i=1}^N$, where $\epsilon_i \sim \mathcal{N}(0, \eta^2)$ is a Gaussian additive noise. Let $\mathbf{f}_* := (f(G_i^*))_{i=1}^{N^*}$ be the values of f at new test locations $\mathbf{G}^* = (G_i^*)_{i=1}^{N^*}$. A zero-mean GP prior is placed on the function f. It follows that the joint distribution of the observed target values and the function values at the test locations writes (see, *e.g.*, Williams and Rasmussen (2006))

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \eta^2 \mathbf{I} & \mathbf{K}_*^T \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix}\right), \tag{1}$$

where $\mathbf{K}$, $\mathbf{K}_{**}$, $\mathbf{K}_*$ are the train, test and test/train Gram matrices, respectively. The posterior distribution of $\mathbf{f}_*$, obtained by conditioning the joint distribution on the observed data, is also Gaussian: $\mathbf{f}_* | \mathbf{G}, \mathbf{y}, \mathbf{G}^* \sim \mathcal{N}(\bar{\mathbf{m}}, \bar{\mathbf{\Sigma}})$ with mean and covariance given by

$$\bar{\mathbf{m}} = \mathbf{K}_*(\mathbf{K} + \eta^2 \mathbf{I})^{-1} \mathbf{y}\,, \tag{2}$$

$$\bar{\mathbf{\Sigma}} = \mathbf{K}_{**} - \mathbf{K}_*(\mathbf{K} + \eta^2 \mathbf{I})^{-1} \mathbf{K}_*^T\,. \tag{3}$$

The mean of the posterior distribution is used as a predictor, and predictive uncertainties can be obtained through the covariance matrix. Figure 2 gives a visual representation of the GP regression for graph inputs.

The predictive performance of GP regression strongly depends on the underlying kernel function k. Most importantly, this kernel function has to be positive definite. In the next section, we introduce our positive definite graph kernel based on optimal transport.

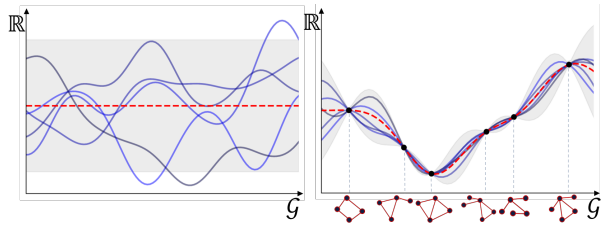

Figure 2: Illustration of Gaussian process for graph inputs. Left: samples from the prior distribution. Right: samples from the posterior distribution after conditioning on observations (input points are graphs here).

4 SWWL GRAPH KERNEL

The proposed methodology relies on the sliced Wasserstein distance (Bonneel et al., 2015) and the continuous Weisfeiler-Lehman (WL) iterations (Togninalli et al., 2019) in order to build a positive definite kernel over graphs. The different steps are detailed after the definition of the kernel.

Definition 1 (SWWL kernel)**.** *Let $P \geqslant 1$ be the number of projections, $Q \geqslant 2$ be the number of quantiles, and $H \geqslant 0$ be the number of WL iterations. The SWWL kernel (illustrated by Figure 1) is defined for any $G, G' \in \mathcal{G}$ by*

$$k_{\mathrm{SWWL}}(G, G') = \exp\left(-\gamma \widehat{SW}_{2,P,Q}^2(\mu_G, \mu_{G'})\right), \tag{4}$$

where $\mu_G = |V|^{-1} \sum_{u \in V} \delta_{\mathbf{E}_u^G}$ is the empirical distribution associated with the continuous WL embedding $\mathbf{E}^G = (\mathbf{E}_u^G)_{u \in V}$ of G with H iterations (see Definition 2) and $\gamma > 0$ is a precision parameter.

Our SWWL kernel has several key properties.

Property 1. *There exists an explicit feature map $\boldsymbol{\phi}$ in a PQ-dimensional space (see Eq. (12)) such that the SWWL kernel can be rewritten as*

$$k_{\mathrm{SWWL}}(G, G') = \exp\left(-\gamma ||\boldsymbol{\phi}(\mu_G) - \boldsymbol{\phi}(\mu_{G'})||_2^2\right). \tag{5}$$

Eq. (5) shows that building the Gram matrix can be broken down into two parts: (1) computing the embeddings $\boldsymbol{\phi}(\mu_G)$ and (2) assembling their pseudo-distance matrix. Importantly, the time complexity of the latter step is independent of the number of graph nodes.

Property 2. *Let δ be the average degree of nodes and n be the average number of nodes. The time complexity for assembling the Gram matrix of the kernel given by Eq. (4) is*

$$\mathcal{O}(NH\delta n + NPn(\log(n) + H) + N^2 PQ)\,.$$

The space complexity is $\mathcal{O}(NPQ)$.

Property 3. *The SWWL kernel is positive definite.*

The proofs are given in the Supplementary Material. We now detail the construction of the feature map $\boldsymbol{\phi}$ involved in Eq. (5).

Continuous Weisfeiler-Lehman embeddings. The WL embeddings were originally proposed for graphs having nodes with discrete labels, but they were extended to continuous attributes by Togninalli et al. (2019).

Definition 2 (Continuous WL embeddings)**.** *Let $G = (V, E, w, \mathbf{F})$ be a graph with attributes $\mathbf{F} = (\mathbf{F}_u)_{u\in V}$, $\mathbf{F}_u \in \mathbb{R}^d$. The continuous Weisfeiler-Lehman iterations are defined recursively for $h \in \mathbb{N}$ by*

$$\mathbf{F}_u^{(h+1)} = \frac{1}{2}\left(\mathbf{F}_u^{(h)} + \frac{1}{\deg(u)} \sum_{v\in\mathcal{N}(u)} w(u,v)\mathbf{F}_v^{(h)}\right), \quad (6)$$

with $\mathbf{F}_u^{(0)} = \mathbf{F}_u$ for $u \in V$. Given a number of iterations $H \geqslant 0$, the continuous WL node embedding of the graph G is the concatenation of the WL iterations at steps $0, 1, \ldots, H$ denoted by $\mathbf{E}^G = (\mathbf{E}_u^G)_{u\in V}$, $\mathbf{E}_u^G \in \mathbb{R}^{(H+1)d}$.

Optimal transport distances. Once the continuous WL embeddings $\mathbf{E}^G$ have been obtained for all graphs, their empirical distributions can be considered and the Wassertein distances can be computed between all pairs of empirical distributions to assemble a kernel. We first recall the definition of the Wasserstein distance for arbitrary measures on $\mathbb{R}^s, s \geqslant 1$, but we will see they do not lead to positive definite kernels.

Definition 3 (Wasserstein distance)**.** *Let $s \geqslant 1$ be an integer, $r \geqslant 1$ be a real number, and μ, ν be two probability measures on $\mathbb{R}^s$ having finite moments of order r. The r-Wasserstein distance is defined as*

$$W_r(\mu,\nu) := \left(\inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathbb{R}^s\times\mathbb{R}^s} ||\mathbf{x}-\mathbf{y}||^r d\pi(\mathbf{x},\mathbf{y})\right)^{\frac{1}{r}}, \quad (7)$$

where $\Pi(\mu,\nu)$ is the set of all probability measures on $\mathbb{R}^s \times \mathbb{R}^s$ whose marginals w.r.t. the first and second variables are respectively μ and ν and $||\cdot||$ stands for the Euclidean norm on $\mathbb{R}^s$.

The Wasserstein distance between two empirical measures can be computed with linear programming in $\mathcal{O}(n^3\log(n))$ or accelerated with entropy regularization (e.g. Sinkhorn regularization (Cuturi, 2013)) in $\mathcal{O}(n^2\log(n))$. Distance substitution kernels (Haasdonk and Bahlmann, 2004) offer a natural way to build kernel functions between empirical distributions. Unfortunately, it is not possible to design positive definite kernels by plugging the Wasserstein distance into such kernels when the dimension s of the space $\mathbb{R}^s$ is greater than one (Peyré et al., 2019). To circumvent this issue, we rely on the sliced Wasserstein distance (Bonneel et al., 2015), which averages one-dimensional Wasserstein distances between the distributions projected on the unit sphere. First, the complexity is reduced because the $1d$-Wasserstein distance can be computed in $\mathcal{O}(n\log(n))$, and second, the corresponding substitution kernel is positive definite.

Definition 4 (Sliced Wasserstein distance)**.** *Let $s \geqslant 1$, $r \geqslant 1$. The r-sliced Wasserstein distance is defined as*

$$SW_r(\mu,\nu) := \left(\int_{\mathbb{S}^{s-1}} W_r(\theta^*_\sharp\mu, \theta^*_\sharp\nu)^r d\sigma(\theta)\right)^{\frac{1}{r}}, \quad (8)$$

where $\mathbb{S}^{s-1}$ is the $(s-1)$-dimensional unit sphere, σ is the uniform distribution on $\mathbb{S}^{s-1}$, $\theta^* : \mathbf{x} \mapsto \langle \mathbf{x}, \boldsymbol{\theta}\rangle$ the projection function of $\mathbf{x} \in \mathbb{R}^s$ in the direction $\boldsymbol{\theta} \in \mathbb{S}^{s-1}$ and $\theta^*_\sharp\mu$ the push-forward measure of μ by θ^*.

In practice, Monte Carlo estimation is performed by sampling P directions $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_P$ uniformly on $\mathbb{S}^{s-1}$, with controlled error bounds (Nietert et al. (2022); Nadjahi et al. (2020)). Although questioned in recent approaches (Deshpande et al. (2019)), Monte Carlo estimation is a common approach that leads to good results in practice. Alternatively, quasi Monte Carlo methods could be used (in particular for low dimensional problems).

Projected quantile embeddings. The computation of the $1d$-Wasserstein distance between empirical measures usually consists in sorting the projected points and then summing a power of the Euclidean distances between values at the same ranks. Here, we propose instead to use $Q \ll n$ equally-spaced quantiles, that do not depend on the distribution sample size n. This strategy differs from usual implementations (Flamary et al., 2021), where the grid of quantiles is chosen according to each pair of input distributions. More precisely, let $0 = t_1 < \cdots < t_\ell < \cdots < t_Q = 1$ be a sequence of Q equally-spaced points in $[0,1]$, and let $\boldsymbol{\theta} \in \mathbb{S}^{s-1}$ be a projection direction. The Q-quantiles estimation of the $1d$-Wasserstein distance between the push forward measures $\mu_\theta := \theta^*_\sharp\mu$ and $\nu_\theta := \theta^*_\sharp\nu$ writes

$$\tilde{W}_{r,Q}(\mu_\theta,\nu_\theta) = \left(\frac{1}{Q}\sum_{\ell=1}^{Q} \left|F^{-1}_{\mu_\theta}(t_\ell) - F^{-1}_{\nu_\theta}(t_\ell)\right|^r\right)^{\frac{1}{r}} \quad (9)$$

where $F_\mu(x) = \mu((-\infty, x]), x \in \mathbb{R}$ is the cumulative distribution function and $F^{-1}_\mu(t) = \inf\{x \in \mathbb{R} :$

$F_\mu(x) \geqslant t\}, t \in [0,1]$ is the inverse cumulative distribution function. The sliced Wasserstein distance given by Eq. (8) is finally estimated as follows:

$$\widehat{SW}_{r,P,Q}(\mu,\nu) = \left(\frac{1}{P}\sum_{p=1}^{P} \tilde{W}_{r,Q}(\mu_{\theta_p},\nu_{\theta_p})^r\right)^{\frac{1}{r}}, \quad (10)$$

where $\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_P$ denote P projection directions uniformly drawn on $\mathbb{S}^{s-1}$. By combining Eqs. (9) and (10), this estimate can be rewritten as

$$\widehat{SW}_{r,P,Q}(\mu,\nu) = ||\boldsymbol{\phi}(\mu) - \boldsymbol{\phi}(\nu)||_r\,, \quad (11)$$

where $||.||_r$ is the r-norm in $\mathbb{R}^{PQ}$, and $\boldsymbol{\phi}$ is the explicit feature map

$$\boldsymbol{\phi}_{p+P(q-1)}(\mu) = (PQ)^{-1/r} F^{-1}_{\mu_{\theta_p}}(t_q) \quad (12)$$

for $p = 1,\dots,P$, $q = 1,\dots,Q$. This feature map is called the *projected quantile embedding*. It provides a PQ-dimensional representation of any probability distribution μ on $\mathbb{R}^s$. In Definition 1 the probability distributions of interest are the empirical distributions associated with the continuous WL embeddings $\mathbf{E}^G = (\mathbf{E}^G_u)_{u\in V}$ of graphs $G \in \mathcal{G}$ with H iterations, so that $s = H(d+1)$. We, therefore, get Property 1.

5 EXPERIMENTS

In our experiments, we consider two different tasks:

- Classification of small graphs with less than 100 nodes corresponding to molecules. This serves as a necessary validation of our kernel with respect to state-of-the-art kernels, and as an illustration of its reduced complexity.
- Gaussian process regression for large graphs from mesh-based simulations in computational fluid dynamics and mechanics. Here we demonstrate that the SWWL kernel can easily handle more than 10^5 nodes in a few seconds or minutes.

Details about the datasets and in particular their graph characteristics are given in Table 1.

5.1 Classification

Experimental setup. The proposed SWWL kernel is compared with 5 existing graph kernels, namely, WWL (Togninalli et al., 2019), Fused Gromov Wasserstein (FGW) (Vayer et al., 2018), Restricted Projected Wasserstein (RPW) (Kaloga et al., 2022), Propagation (PK) (Neumann et al., 2016) and Graph Hopper (GH) (Feragen et al., 2013). Five datasets are taken from the TUD benchmark (Morris et al., 2020): BZR, COX2, PROTEINS, ENZYMES, Cuneiform. We only keep continuous attributes for the nodes and ignore categorical information except for Cuneiform where we use the same setup as proposed by Kaloga et al. (2022) with fused attributes.

Following the same approach as in previous work, we use a SVM classifier with the (indefinite) kernel $k(G,G') := \exp(-\gamma\mathcal{D}(G,G'))$ where $\mathcal{D}$ is the (pseudo)distance defined by the method at hand. There are 3 hyperparameters to set: the number of WL iterations $H \in \{0,1,2,3\}$, the kernel precision $\gamma \in \{10^{-4},10^{-3},\dots,10^{-1}\}$, and the SVM regularization parameter $C \in \{10^{-3},10^{-2},\dots,10^{3}\}$. In the case of the SWWL kernel, we arbitrarily set the number of projections to $P = 20$, and the number of quantiles to $Q = 20$. It has been found that their influence is negligible on these small datasets. A more detailed study in the case of regression tasks with large graphs is carried out in Section 5.2.

To train and evaluate all baselines, we rely on nested cross-validation as in Togninalli et al. (2019). We use stratified 10-fold train/test split in the outer loop. For each split, the model hyperparameters are tuned by a cross-validation scheme with an inner stratified 5-fold defined on the current train set. We then compute the accuracy for the current test set and finally average the results obtained for all the 10 outer splits. The same random splits and seeds are used for all the considered methods in order to ensure reproductibility and fair comparisons. Results are reported in Table 2.

Discussion about the results. As Table 2 shows, our SWWL kernel achieves similar results for graph classification as other state-of-the-art kernels. It is worth emphasizing again that the SWWL kernel is not intended to outperform other methods here, but rather to be a positive-definite variant of the WWL kernel with drastically reduced complexity. Remark that due to the small sample size and high class imbalance in these datasets, the results strongly depend on the initial train/test split, thus explaining the observed high variance.

Time-based comparisons of methods are given in Table 3. They should be taken with hindsight, especially when considering that they do not all rely on the same backend. Computation times for the RPW kernel are taken from the original paper, since our own implementation was much slower. In summary, it can be seen that the SWWL kernel largely outperforms other methods in terms of computation times thanks to its explicit unsupervised embeddings.

5.2 Regression

We now apply the SWWL kernel to three challenging high-dimensional regression problems from computa-

Table 1: Summary of the datasets. (∗): fixed number of nodes and adjacency structure, CM: coarsened meshes.

Dataset	Num graphs	Mean nodes	Mean edges	Attributes	Scalars	Task (classes)
BZR	405	35.7	38.4	3	0	Classif(2)
COX2	467	41.2	43.5	3	0	Classif(2)
PROTEINS	1113	39.1	72.8	1	0	Classif(2)
ENZYMES	600	32.6	62.1	18	0	Classif(6)
Cuneiform	267	21.27	44.8	3 (+2)	0	Classif(30)
Rotor37*	1000+200	29773	77984	3	2	Regression
Rotor37-CM	1000+200	1053.8	3097.4	3	2	Regression
Tensile2d	500+200	9425.6	27813.8	2	6	Regression
Tensile2d-CM	500+200	1177.4	3159.8	2	6	Regression
AirfRANS	800+200	179779.0	536826.6	2	2	Regression
AirfRANS-CM	800+200	6852.8	19567.2	2	2	Regression

tional fluid and solid mechanics.

Rotor37 dataset. The NASA rotor 37 case serves as a prominent example of a transonic axial-flow compressor rotor widely employed in computational fluid dynamics research. This dataset is made of 3D compressible steady-state Reynold-Averaged Navier-Stokes (RANS) simulations that model external flows (Ameri, 2009). The inputs of the simulations are given by the finite element mesh of a 3D compressor blade (graphs with ~ 30000 nodes), and two scalar physical parameters corresponding to the rotational speed and the input pressure. The scalar output of the problem is the isentropic efficiency, which is computed by solving the boundary value problem with the finite volume method using the `elsA` software (Cambier et al., 2011).

Tensile2d dataset. We consider a two-dimensional problem in solid mechanics introduced by Casenave et al. (2023). The input geometries consist of 2D squares with two half circles that have been cut off in a symmetrical manner. Pieces are subject to a uniform pressure field over the upper boundary and the material of the structures is modeled by a nonlinear elasto-viscoplastic law. The inputs of the problem are given by the mesh of the geometry (graphs with ~ 10000 nodes), and six scalar parameters that correspond to the material parameters and the input pressure applied to the upper boundary. The scalar output of the problem is the maximum Von Mises stress across the domain, computed numerically with the finite element method using the `Zset` software Garaud et al. (2019).

AirfRANS dataset. This dataset consists of RANS simulations over 2D NACA airfoils with geometric variabilities in a subsonic flight regime (Bonnet et al., 2022). The inputs of the problem are given by the finite element mesh of the domain surrounding the airfoil (graphs with more than 10^5 nodes), and two scalar inputs, the angle of attack and the inlet velocity. The scalar output of the problem is the drag coefficient. More details about the distributions of the training and test sets can be found in the work of Bonnet et al. (2022).

As summarized in Table 1, the input meshes are made of a large number of nodes. While our SWWL kernel is able to deal with such large graphs, we have observed numerically that no alternative graph kernel is tractable in this setting. For this reason, we also consider a degraded version of each dataset with a largely reduced number of nodes. Such smaller dimensional problems are obtained with coarsened meshes (CM) using the MMG remesher (Balarac et al., 2022), while the scalar inputs and outputs are unchanged. The associated datasets are referred to

Table 2: Mean accuracy (%) and standard deviation for 10 experiments of classification of small graphs.

	Kernel/Dataset	BZR	COX2	PROTEINS	ENZYMES	Cuneiform
OT-based	SWWL (ours)	**85.43 ± 4.05**	78.61 ± 5.87	**75.12 ± 5.99**	66.67 ± 5.0	83.36 ± 4.32
	WWL	84.43 ± 2.82	75.62 ± 6.43	74.85 ± 4.97	**70.33 ± 2.87**	84.62 ± 6.78
	FGW	**85.41 ± 3.14**	76.05 ± 7.98	71.79 ± 3.61	67.83 ± 2.36	80.85 ± 8.06
	RPW	**85.42 ± 2.41**	77.98 ± 5.54	71.42 ± 5.10	52.0 ± 6.94	**91.00 ± 8.36**
Non-OT-based	PK	80.96 ± 4.79	78.21 ± 7.41	69.54 ± 4.90	68.5 ± 5.13	-
	GH	82.44 ± 4.98	**79.49 ± 6.04**	71.97 ± 2.44	43.5 ± 3.91	-

Table 3: Computation times (s) needed to build the distance/Gram matrices of small graphs for all hyperparameters. When embeddings are pre-computed, we give both embedding times + distance times.

	Kernel/Dataset	BZR	COX2	PROTEINS	ENZYMES	Cuneiform
OT-based	SWWL (ours)	**0.7 + 0.1**	**0.6 + 0.1**	**1.5 + 0.6**	**1.1 + 0.2**	**1.3 + 0.1**
	WWL	0.3 + 97	0.3 + 131.2	0.7 + 803	0.5 + 220	0.9 + 34
	FGW	0.6 + 714	0.7 + 842	1.6 + 7882	0.9 + 1381	0.4 + 145
	RPW	35 + 5	40 + 7	240 + 40	220 + 40	-
Non-OT-based	PK	10	13	52	53	-
	GH	77	108	3998	230	-

`Rotor37-CM`, `Tensile2d-CM`, and `AirfRANS-CM`. Figure 3 shows samples from the datasets and their coarsened variants.

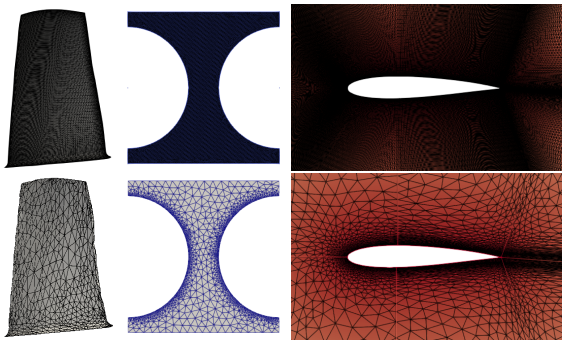

Figure 3: Top row: meshes from the `Rotor37`, `Tensile2d` and `AirfRANS` datasets (from left to right). Bottom row: coarsened versions of the meshes.

Choice of the number of iterations. Usually it is recommended to choose a small number of iterations to avoid an oversmoothing phenomenon (Xu et al. (2018)), but the impact of this hyperparameter is largely overlooked in the literature. Since we work here with large graphs, we propose to investigate the incorporation of skips during the WL iterations in order to visit larger neighborhoods. The step T is chosen to be approximately $\sqrt{n}$ and we keep only iterations 0, T, $2T$ and $3T$: the resulting kernel is called SWWL($\sqrt{n}$) in the following, in contrast to SWWL(1) for the original one with iterations $0, \dots, 3$. However note that computing iterations with $T = O(\sqrt{n})$ changes the complexity of SWWL embeddings from $\mathcal{O}(NH\delta n + NPn\log(n))$ to $\mathcal{O}(N\delta n\sqrt{n} + NPn\log(n))$. The default number of projections for both kernels is set to 50 and the default number of quantiles to 500.

Experimental setup. For all the considered problems, the train and test sets have been generated by a space-filling design of experiments, including for the inputs given as graphs, since the mesh geometry is actually parameterized beforehand. The training of the GP is repeated 5 times, and we report the mean and standard deviations of the evaluation metrics. The kernel function of the GP is chosen as a tensorized kernel of the form

$$k(\mathfrak{X}, \mathfrak{X}') := \sigma^2 k_{\mathrm{SWWL}}(G, G') \prod_{\ell=1}^{m} c_{5/2}(|s_\ell - s'_\ell|) \quad (13)$$

where $\mathfrak{X} = (G, s_1, \dots, s_m)$ and $\mathfrak{X}' = (G', s'_1, \dots, s'_m)$ denote the input graphs and the m input scalars $s_1, \dots, s_m$, $c_{5/2}$ is the Matérn-5/2 covariance function, and σ^2 is a variance parameter. The lengthscale parameters of the SWWL and Matérn-5/2 kernels are optimized simultaneously by maximizing the marginal posterior log-likelihood following the work of Gu et al. (2018) (see the Supplementary Material).

Discussion about the results. We first observe in Table 4 that adding the information of skipped iterations decreases the RMSE on `AirfRANS-CM` and `AirfRANS`, but it does not seem to significantly improve performance on other datasets. Such a decrease in RMSE however does not seem to justify the important rise in computation time for the embeddings. Second, we achieve lower RMSE for SWWL than for WWL and PK on all CM datasets. Table 5 also illustrates that SWWL outperforms other methods in terms of computation times even when the number of WL iterations is large. But on these coarsened datasets, even though they make other kernels tractable, we achieve much lower accuracies when compared to the original ones, meaning that the naive coarsening strategy is not efficient. Taking the complete data and doing an efficient dimension reduction with SWWL is undoubtedly a better strategy.

Finally, we also investigate the impact of the number of projections and the number of quantiles on the SWWL kernel. Figure 4 shows that on the `Rotor37` dataset, the mean RMSE decreases and the standard deviation diminishes when the number of projections and quantiles grow, as expected. However, above 20 projections and 100 quantiles the RMSE score decreases very little, and we observe the same behavior for all other

Table 4: Mean RMSE and standard deviation for 5 experiments of Gaussian process regression. Missing values correspond to intractable experiments due to memory and/or CPU usage.

Kernel/Dataset	Rotor37 x10^{-3}	Rotor37-CM x10^{-3}	Tensile2d x1	Tensile2d-CM x1	AirfRANS x10^{-4}	AirfRANS-CM x10^{-4}
SWWL(1)	1.44 ± 0.07	**3.49 ± 0.15**	**0.89 ± 0.01**	**1.51 ± 0.01**	7.56 ± 0.36	9.63 ± 0.54
SWWL($\sqrt{n}$)	**1.36 ± 0.05**	3.64 ± 0.10	**0.90 ± 0.01**	1.57 ± 0.03	**7.29 ± 0.41**	**8.07 ± 0.34**
WWL	-	**3.51 ± 0.00**	-	6.46 ± 0.00	-	14.4 ± 0.80
PK	-	4.18 ± 0.39	-	6.03 ± 4.58	-	8.94 ± 2.31

Table 5: Embedding and distance computation times needed for a single Gram matrix. When embeddings are pre-computed, we give both embedding times + distance times. (*): Parallelized over 100 processes.

Kernel/Dataset	Rotor37	Rotor37-CM	Tensile2d	Tensile2d-CM	AirfRANS	AirfRANS-CM
SWWL(1)	**1min + 11s**	**4s + 11s**	**11s + 4s**	**2s + 4s**	**5min + 7s**	**15s + 7s**
SWWL($\sqrt{n}$)	54min + 11s	20s + 11s	5min + 4s	10s + 4s	3h52min + 7s	7min + 7s
WWL	-	13min (*)	-	6min (*)	-	8h (*)
PK	-	1min	-	2min	-	15min

datasets. This means that in general an embedding of size 20×100 is sufficient to achieve good performance.

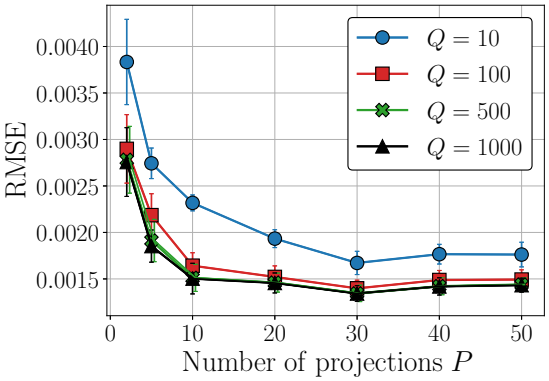

Figure 4: Impact of number of projections and quantiles on RMSE for GP regression for Rotor37.

Implementation and computing infrastructure. We leverage an available Python implementation of continuous WL iterations (Fey and Lenssen, 2019), and GP regression in R (Gu et al., 2018). The FGW distances are computed with the POT library (Flamary et al., 2021). The PK and GH kernels are implemented in the Grakel library (Siglidis et al., 2020), and we use the original implementation of RPW (Kaloga et al., 2022). All our analyses were performed on a shared server running Linux CentOS 7.9.2009, with 1 CPU (Intel Xeon Gold 6342 @ 2.80GHz) using 4 cores, and a total of 21,6 GB of RAM.

6 CONCLUSION

The Sliced Wasserstein Weisfeiler-Lehman is a new positive definite graph kernel with drastically reduced complexity compared to existing graph kernels. It is more efficient for graphs where attributes are of paramount importance compared to the adjacency structure. We demonstrate its efficiency, with equivalent performance to state-of-the-art kernels for common benchmarks of small molecules. We also propose for the first time Gaussian process regression of large-scale meshes used in computational physics where the size of graphs usually limits the use of existing kernels. Our kernel can be viewed as an explicit unsupervised embedding, with complexity related to the number of projections and quantiles used in the sliced Wasserstein distance. Our experiments show that few of them are actually sufficient to encapsulate information, thus achieving an explicit and efficient dimension reduction.

We only considered here supervised learning tasks, but our kernel can be plugged into any other kernel-based method, and may thus be of particular interest for graph clustering with kernel k-means, or space-filling designs with the maximum mean discrepancy.

A remaining question is the impact of the unsupervised WL iterations. A more supervised version with an optimization phase may help increase the representation capacity, at the expense of non-negligible additional training cost, and such trade-off should be further studied. Finally, our kernel can only handle scalar outputs, and future work may include an extension to vector fields, since in computational physics the outputs obtained by simulation are often of this form.

References

Ameri, A. (2009). Nasa rotor 37 cfd code validation glenn-ht code. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 1060.

Balarac, G., Basile, F., Bénard, P., Bordeu, F., Chapelier, J.-B., Cirrottola, L., Caumon, G., Dapogny, C., Frey, P., Froehly, A., et al. (2022). Tetrahedral remeshing in the context of large-scale numerical simulation and high performance computing. *MathematicS In Action*, 11(1):129–164.

Bonneel, N., Rabin, J., Peyré, G., and Pfister, H. (2015). Sliced and radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51:22–45.

Bonnet, F., Mazari, J., Cinnella, P., and Gallinari, P. (2022). Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478.

Borgwardt, K., Ghisu, E., Llinares-López, F., O'Bray, L., Rieck, B., et al. (2020). Graph kernels: State-of-the-art and future challenges. *Foundations and Trends® in Machine Learning*, 13(5-6):531–712.

Cambier, L., Gazaix, M., Heib, S., Plot, S., Poinot, M., Veuillot, J., Boussuge, J., and Montagnac, M. (2011). An overview of the multi-purpose elsa flow solver. *Aerospace Lab*, (2):p–1.

Casenave, F., Staber, B., and Roynard, X. (2023). Mmgp: a mesh morphing Gaussian process-based machine learning method for regression of physical problems under non-parameterized geometrical variability. *arXiv preprint arXiv:2305.12871*.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

Deshpande, I., Hu, Y.-T., Sun, R., Pyrros, A., Siddiqui, N., Koyejo, S., Zhao, Z., Forsyth, D., and Schwing, A. G. (2019). Max-sliced Wasserstein distance and its use for gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10648–10656.

Feragen, A., Kasenburg, N., Petersen, J., de Bruijne, M., and Borgwardt, K. (2013). Scalable kernels for graphs with continuous attributes. *Advances in neural information processing systems*, 26.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. (2021). Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8.

Garaud, J.-D., Rannou, J., Bovet, C., Feld-Payet, S., Chiaruttini, V., Marchand, B., Lacourt, L., Yastrebov, V., Osipov, N., and Quilici, S. (2019). Z-set-suite logicielle pour la simulation des matériaux et structures. In *14ème Colloque National en Calcul des Structures*.

Gramacy, R. B. (2020). *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. CRC press.

Gu, M., Wang, X., and Berger, J. O. (2018). Robust Gaussian stochastic process emulation. *The Annals of Statistics*, 46(6A):3038–3066.

Haasdonk, B. and Bahlmann, C. (2004). Learning with distance substitution kernels. In *Joint pattern recognition symposium*, pages 220–227. Springer.

Haussler, D. et al. (1999). Convolution kernels on discrete structures. Technical report, Citeseer.

Hein, M. and Bousquet, O. (2005). Hilbertian metrics and positive definite kernels on probability measures. In *International Workshop on Artificial Intelligence and Statistics*, pages 136–143. PMLR.

Kaloga, Y., Borgnat, P., and Habrard, A. (2022). A simple way to learn metrics between attributed graphs. In *Learning on Graphs Conference*, pages 25–1. PMLR.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440.

Kolouri, S., Naderializadeh, N., Rohde, G. K., and Hoffmann, H. (2020). Wasserstein embedding for graph learning. *arXiv preprint arXiv:2006.09430*.

Kriege, N. M., Johansson, F. D., and Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, 5(1):1–42.

Meunier, D., Pontil, M., and Ciliberto, C. (2022). Distribution regression with sliced Wasserstein kernels. In *International Conference on Machine Learning*, pages 15501–15523. PMLR.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. (2020). Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

Morris, C., Kriege, N. M., Kersting, K., and Mutzel, P. (2016). Faster kernels for graphs with continuous attributes via hashing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1095–1100. IEEE.

Nadjahi, K., Durmus, A., Chizat, L., Kolouri, S., Shahrampour, S., and Simsekli, U. (2020). Statistical and topological properties of sliced probability divergences. *Advances in Neural Information Processing Systems*, 33:20802–20812.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. (2016). Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, 102:209–245.

Nietert, S., Goldfeld, Z., Sadhu, R., and Kato, K. (2022). Statistical, robustness, and computational guarantees for sliced Wasserstein distances. *Advances in Neural Information Processing Systems*, 35:28179–28193.

Nikolentzos, G., Siglidis, G., and Vazirgiannis, M. (2021). Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027.

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. (2020). Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*.

Siglidis, G., Nikolentzos, G., Limnios, S., Giatsidis, C., Skianis, K., and Vazirgiannis, M. (2020). Grakel: A graph kernel library in python. *The Journal of Machine Learning Research*, 21(1):1993–1997.

Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., and Borgwardt, K. (2019). Wasserstein Weisfeiler-Lehman graph kernels. *Advances in neural information processing systems*, 32.

Vayer, T., Chapel, L., Flamary, R., Tavenard, R., and Courty, N. (2018). Optimal transport for structured data with application on graphs. *arXiv preprint arXiv:1805.09114*.

Wijesinghe, A., Wang, Q., and Gould, S. (2021). A regularized Wasserstein framework for graph kernels. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 739–748. IEEE.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*. MIT press.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Material for Gaussian process regression with Sliced Wasserstein Weisfeiler-Lehman graph kernels

A Proofs

A.1 Positive definiteness of SWWL

In this section we give a proof of Property 3 (The SWWL kernel is positive definite). We first recall useful properties about Hilbertian pseudo-distances and their link with positive definite kernels.

Definition 5 (Hilbertian (pseudo)-distance)**.** *((Hein and Bousquet, 2005)) A pseudo-distance d defined on a set X is said to be* Hilbertian *if there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : X \to \mathcal{H}$ such that for any pair x, x' in X, $d(x, x') = ||\phi(x) - \phi(x')||_{\mathcal{H}}$.*

Pseudo means that d might not satisfy $d(x, x') = 0$ implies $x = x'$. All usual kernels substituted with distances are positive definite if and only if the distance is Hilbertian as the following property shows.

Property 4. *Meunier et al. (2022) Introducing $\langle x, x' \rangle_d^{x_0} = \langle \phi(x) - \phi(x_0), \phi(x') - \phi(x_0) \rangle_{\mathcal{H}}$, the following statements are equivalent:*

- *d is a Hilbertian pseudo-distance.*
- *$k_{lin}(x, x') = \langle x, x' \rangle_d^{x_0}$, $(x, x') \in X^2$ is positive definite for all $x_0 \in X$.*
- *$k_{poly}(x, x') = (c + \langle x, x' \rangle_d^{x_0})^l$, $(x, x') \in X^2$ is positive definite for all $x_0 \in X$, $c \geqslant 0$, $l \in \mathbb{N}$.*
- *$k(x, x') = \exp(-\gamma d^{2\beta}(x, x'))$, $(x, x') \in X^2$ is positive definite for all $\gamma \geqslant 0$, $\beta \in [0, 1]$.*

It is possible to show that Wasserstein distances are not Hilbertian, but that sliced Wasserstein distances are Hilbertian.

Property 5. *(Peyré et al. (2019)) $\sqrt{W_1}$ and W_2 are not Hilbertian distances in $\mathbb{R}^s$ when $s \geqslant 2$.*

Property 6. *(Meunier et al. (2022)) $\sqrt{SW_1}$ and SW_2 are Hilbertian distances in $\mathbb{R}^s$ for all $s \geqslant 2$.*

We can now prove the positive definiteness of SWWL.

Proof. We recall the form of the estimated sliced Wasserstein distance between two distributions μ and ν with P projections and Q quantiles: $\widehat{SW}_{r,P,Q}(\mu, \nu) = ||\phi(\mu) - \phi(\nu)||_r$ where $||.||_r$ is the r-norm in $\mathbb{R}^{PQ}$ and $\boldsymbol{\phi}_{p+P(q-1)}(\mu) = (PQ)^{-1/r} F^{-1}_{\mu_{\theta_p}}(t_q)$ for $p = 1, \ldots, P$ and $q = 1, \ldots, Q$. $\widehat{SW}_{r,P,Q}$ is a Hilbertian pseudo-distance 5 as we built an explicit feature map ϕ to $\mathcal{H} = \mathbb{R}^{PQ}$.
If we deal with empirical measures with maximal support size $\hat{n}$ (supposed to be finite), $\widehat{SW}_{r,P,Q}$ is not necessarily a distance when $Q < \hat{n}$ as points from some measures can be lost by retaining only Q quantiles.

Using Property 4, we obtain that: $\mu, \nu \mapsto \exp\left(-\gamma \widehat{SW}^2_{2,P,Q}(\mu, \nu)\right)$ is a positive definite kernel for $\gamma > 0$.

Finally, this property still holds if we restrict the space of measures to be $\{\mu_G = |V|^{-1} \sum_{u \in V} \delta_{\mathbf{E}^G_u}, G \in \mathcal{G}\}$, i.e. the empirical distributions associated with the continuous WL embeddings of graphs in $\mathcal{G}$.

We conclude that for $\gamma > 0$, $k_{\text{SWWL}}(G, G') = \exp\left(-\gamma \widehat{SW}^2_{2,P,Q}(\mu_G, \mu_{G'})\right)$ is a positive definite kernel. □

If we take $Q \geqslant max\{|V|, G \in \mathcal{G}\}$ and $P \to +\infty$, $\widehat{SW}_{r,P,Q}$ is then exactly the sliced Wasserstein distance for empirical measures.

Remark that our approximation of sliced Wasserstein distances is different from the one proposed in Meunier et al. (2022). We actually used a fixed number of quantiles Q and we look at equally spaced values, whereas for them positions are drawn uniformly at random, which does not guarantee a complete recovery of the original distribution even if the number of draws is equal to its size.

A.2 Complexity of SWWL

In this section we give a proof of Property 2 (Complexity of SWWL).

Proof. N denotes the number of graphs, n the average number of nodes, P the number of projections and Q the number of quantiles.

The first step is the computation of WL iterations. For a graph of size n, one WL iteration propagates information of nodes to their neighbours in $O(n\delta)$ where δ is the mean number of neighbours. Performing H iterations for all the N graphs is done in $O(NH\delta n)$. The next step is to compute all P projections for all graphs in $O(NPHn)$ and then to sort the 1-dimensional projected embeddings to obtain the Q quantiles, which can be done for all graphs in $O(NPn\log(n))$. Once projected quantile embeddings are computed for all graphs, the kernel is built by taking all pairwise distances between the embeddings in $O(N^2PQ)$ (see Figure 5).
We conclude that the time complexity of SWWL is $O(NH\delta n + NPn(\log(n) + H) + N^2PQ)$.

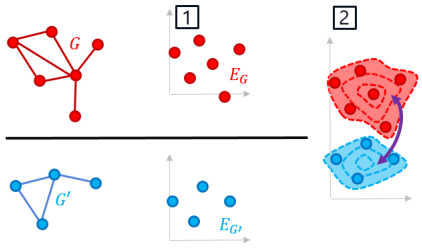

Figure 5: Distance between embeddings of graphs seen as distributions. The black line represents operations that can be performed separately on each input $G_1, \ldots, G_N$. Step 1: graph embedding. Step 2: distance between distributions.

For the spatial complexity, we need to store all the projected quantile embeddings in $O(NPQ)$. Remark that in theory, all embeddings do not need to be loaded simultaneously. During the computation of the distance matrix, we could only load 2 graphs in memory each time a distance is computed, resulting in a space complexity of $O(PQ)$. In practice, it is more convenient to keep all embeddings in memory in order to compute pairwise distances required for the Gram matrix at once. □

In comparison, using the Wasserstein distance would lead to a $O(N^2n^3log(n))$ time complexity and a $O(n^2)$ space complexity to build the Gram matrix, which can be prohibitive when the number of nodes is large.

B Details about the compared methods and their hyperparameters

In this section we give more details on hyperparameters of reference methods for classification and regression tasks presented in the experimental section. We also explain the choice of the compared methods for regression. In particular, we emphasize the difficulties of applying other approaches to the large meshes of interest, hence we also consider coarsened meshes which make it possible to compare the different methods.

B.1 Hyperparameter grids for classification of small graphs.

In this section we give some additional details about the hyperparameter tuning for the SVM classification of small graphs.

FGW: The balance parameter α between structure and features is tuned in $[0, 0.25, 0.5, 0.75, 1.0]$.

SGML: We use the same setting as in Kaloga et al. (2022). For the SGCN part, we use 10 iterations and a learning rate of $1e^{-2}$ (except for PROTEINS with 20 iterations, a learning rate of $1e^{-3}$ and ENZYMES with 20 iterations, a learning rate of $1e^{-4}$). The number of layers is optimized in $[1, 2, 3]$. The final layer dimension is fixed to 5. For the Restricted Wasserstein part, we take 50 uniformly sampled projections.

GH: We use a RBF kernel to compare node attributes. The current implementation of Grakel (Siglidis et al. (2020)) uses a fixed lengthscale parameter of 1.

PK: The results are given for a bin width $w = 1e^{-2}$ and we optimize the number of iterations $[1, 3, 5, 8, 10, 15, 20]$. We have also tested with varying bin width $w \in \{1e^{-5}, 1e^{-4}, \dots, 1e^{-1}\}$ but we have found that $w = 1e^{-2}$ gives the best scores.

B.2 Reference methods for regression of large meshes.

Two major problems arise when comparing to WWL. The time to calculate a Wasserstein distance between embeddings of size 30000 is approximately 80 seconds, so more than 400 days are needed to compute the pairwise distances of 1000 graphs. The second issue is memory usage. Allocating matrices of size 30000×30000 for the transport costs of the Wasserstein distance is only possible with a minimum amount of space and severely limits parallelization. We still include a comparison with WWL on coarsened datasets with 100 parallel jobs as it is the method on which SWWL is based. Remark that we do not use entropic regularization (Cuturi (2013)) that would reduce the complexity but has the (not negligible) cost of an extra hyperparameter to tune in an outer loop.

We do not include FGW and GH for time and memory issues. The custom loss of SGML designed for classification makes it not applicable for regression tasks.

For the special case of PK, we face memory issues. This is mainly due to the transition matrices of all the graphs that are stored simultaneously in the implementation of Siglidis et al. (2020). For the special case of `AirfRANS-CM`, we use a shared server running Linux CentOS 7.9.2009, with 2 CPU (RAMIntel Xeon E5-2680 @ 2.50 GHz) using 24 cores, and a total of 378 GB of RAM. Remark that this computing infrastructure is not sufficient to build the PK kernel with `Rotor37`, `Tensile2d` and `AirfRANS`.

We tested the PK with a bin width $w \in \{1e^{-5}, 1e^{-4}, \dots, 1e^{-1}\}$ and a number of iterations $t_{max} \in \{1, 3, 5, 7\}$. The results given in the paper correspond to $t_{max} = 1$ and $w = 1e^{-2}$ except for `Rotor37-CM` where $w = 1e^{-3}$ as such parameters give the best test results. Note that these parameters should be finely tuned using a hyperparameter grid, thus drastically increasing the times in Table 5.

For $SWWL(\sqrt{n})$, the step of WL iterations is chosen to be respectively $T = 173, 30, 100, 30, 100, 100$ for `Rotor37`, `Rotor37-CM`, `Meca2d`, `Meca2d-CM`, `AirfRANS`, `AirfRANS-CM`. These values correspond approximatively to the square root of the average number of nodes, except for AirFRANS where it is lower for computational reasons. They have therefore not been optimized.

B.3 Robust Gaussian process regression

Several strategies can be used to estimate the hyperparameters in Gaussian process regression. We use the robust Gaussian stochastic process emulation (RGaSP) (Gu et al. (2018)) that estimates robustly the different lengthscale parameters. We first describe the method, we then give implementation details and we finally give the computation times (independent of the choice of the kernel).

Robust estimation of range parameters. We consider a Gaussian process $f \sim \mathcal{GP}(m, k)$ where $m : \mathcal{X} \to \mathbb{R}$ is the mean function and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the covariance function. We write $k(\cdot, \cdot) = \sigma^2 c(\cdot, \cdot)$ to separate the variance σ^2 and the correlation function c. In this section, we suppose that m is a constant function determined by the parameter $\theta \in \mathbb{R}$. The correlation function has range parameters that we denote by $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_L)$ in an anisotropic setting (in our paper $\mathcal{X}$ is a set of graphs and $L = 1$).

We assume that we have observations $y_i = f(\mathbf{x}_i)$ for $i \in \{1, \dots, N\}$ of f at input locations $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N$. We consider the correlation matrix $\mathbf{R} \in \mathcal{M}_{N,N}(\mathbb{R})$ at input points where $(\mathbf{R})_{ij} = c(\mathbf{x}_i, \mathbf{x}_j)$. Following the notations of section 4, the train Gram matrix is thus $\mathbf{K} = \sigma^2 \mathbf{R}$. We omit the white noise that only changes the train

covariance matrices.

The common way to deal with the mean and variance parameters is to use their closed form in a Bayesian setting. These parameters are thus assigned to the objective prior $\pi(\theta,\sigma^2)\propto\frac{1}{\sigma^2}$. This prior is then used to marginalize out the mean and variance parameters in the likelihood function and lead to the marginal likelihood

$$\mathcal{L}(\boldsymbol{\gamma}|\mathbf{X},\mathbf{y})\propto|\mathbf{R}|^{-\frac{1}{2}}|\mathbf{h}^T\mathbf{R}^{-1}\mathbf{h}|^{-\frac{1}{2}}(S^2)^{-\frac{n-1}{2}}, \tag{14}$$

where $\mathbf{h}$ is the $N\times 1$ column vector where all coefficients are equal to 1, $S^2 = \mathbf{y}^T\mathbf{R}^{-1}\left[\mathbf{I_N}-\mathbf{h}(\mathbf{h}^T\mathbf{R}^{-1}\mathbf{h})^{-1}\mathbf{h}^T\mathbf{R}^{-1}\right]\mathbf{y}$. However, the maximum marginal likelihood estimators obtained by maximizing (14) are not robust in the sense that they can lead to covariance matrices close to the identity or to a matrix made up of one, which give rise to numerical issues.

In the robust Gaussian process framework, the marginal likelihood is augmented by a prior for the range parameters $\pi(\theta,\sigma^2,\boldsymbol{\gamma})\propto\frac{\pi(\gamma)}{\sigma^2}$. We use the prior $\pi^{JR}(\boldsymbol{\gamma}) = \left(\sum_{l=1}^L C_l\gamma_l^{-1}\right)^a\exp\left(-b\sum_{l=1}^L C_l\gamma_l^{-1}\right)$, $a=0.2$, $b=N^{-1/L}(a+L)$, and C_l equal to the mean of $|x_{il}-x_{il}|$ for $1\leqslant i,j\leqslant N, i\neq j$. Remark that in the case where x_{il} is not a scalar but rather a structured object, we replace the absolute value by the user-defined distance or pseudo-distance that is substituted in the kernel ((in our paper we use the estimated sliced Wasserstein distance).

The range parameters are estimated by maximization of the marginal posterior distribution

$$(\hat{\gamma_1},\dots,\hat{\gamma_L}) = \underset{\gamma_1,\dots,\gamma_L}{\operatorname{argmax}}(\mathcal{L}(\gamma|\mathbf{X},\mathbf{y})\pi^{JR}(\boldsymbol{\gamma})). \tag{15}$$

We now want to predict unknown targets $\mathbf{f}_* := (f(\mathbf{x}_i^*))_{i=1}^{N^*}$ at N^* new locations $\mathbf{X}^* = (\mathbf{x}_i^*)_{i=1}^{N^*}$. With the use of the estimated range parameters, the predictive distribution of $\mathbf{f}^*$ given $\mathbf{X}$, $\mathbf{y}$ and $\boldsymbol{\gamma}$ is a Student's t-distribution with $N-1$ degrees of freedom

$$\mathbf{f}_*|\mathbf{X},\mathbf{y},\mathbf{X}^*,\gamma\sim t(\bar{\mathbf{m}},\hat{\sigma}^2\bar{\mathbf{C}},N-1), \tag{16}$$

where

$$\bar{\mathbf{m}} = \mathbf{h}_*\hat{\theta}+\mathbf{R}_*\mathbf{R}^{-1}(\mathbf{y}-\mathbf{h}\hat{\theta}), \tag{17}$$

$$\hat{\sigma}^2 = (N-1)^{-1}(\mathbf{y}-\mathbf{h}\hat{\theta})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{h}\hat{\theta}), \tag{18}$$

$$\bar{\mathbf{C}} = \mathbf{R}_{**}-\mathbf{R}_*\mathbf{R}^{-1}\mathbf{R}_*^T+(\mathbf{h}_*-\mathbf{h}^T\mathbf{R}^{-1}\mathbf{R}_*^T)^T(\mathbf{h}^T\mathbf{R}^{-1}\mathbf{h})^{-1}(\mathbf{h}_*-\mathbf{h}^T\mathbf{R}^{-1}\mathbf{R}_*^T), \tag{19}$$

with $\mathbf{h}_*$ being the $N^*\times 1$ vector composed of ones, $\hat{\theta} = (\mathbf{h}^T\mathbf{R}^{-1}\mathbf{h})^{-1}\mathbf{h}^T\mathbf{R}^{-1}\mathbf{y}$ being the least squares estimator for θ, and $\mathbf{R}_*$, $\mathbf{R}_{**}$ being the covariance matrices evaluated respectively between train inputs and test/train inputs.

Computation times. We add the times required to train the Gaussian process regression with RobustGaSP in R given precomputed distance matrices. For `Rotor37` (1000 train inputs), the training takes 220 seconds, for `Tensile2d` (500 train inputs), for `AirfRANS` (800 train inputs) the training takes 135 seconds. This time is similar for coarsened meshes as the shape of distance matrices is the same.

C Additional advantages from GP regression

We finally discuss some benefits coming from GP regression that prove useful in practice for physics-based learning problems.

C.1 Anisotropic variant of SWWL

The difficulty to choose the number of WL iterations has led us to define an anisotropic variant of SWWL. The idea is to leverage the information of each iteration separately by using $H+1$ tensorized SWWL kernels.

Definition 6. *Let H be a number of WL iterations and $G\in\mathcal{G}$ be a graph with its continuous WL embedding $\mathbf{E}^G = (\mathbf{E}_u^G)_{u\in V}$, $\mathbf{E}_u^G\in\mathbb{R}^{(H+1)d}$. We denote by $\mathbf{E}^{G,(h)} = (\mathbf{E}_u^{G,(h)})_{u\in V}$, $\mathbf{E}_u^{G,(h)}\in\mathbb{R}^d$, its h-th continuous WL iteration and by $\mu_G^{(h)} = |V|^{-1}\sum_{u\in V}\delta_{(\mathbf{E}^{G,(h)})_u}$ the associated empirical measure for $h\in\{0,\dots,H\}$.*

The ASWWL kernel is defined as follows:

$$k_{ASWWL}(G, G') = \prod_{h=0}^{H} \exp\left(-\gamma_h \widehat{SW}^2_{2,P,Q}(\mu_G^{(h)}, \mu_{G'}^{(h)})\right), \tag{20}$$

for $\gamma_0 > 0, \gamma_1 > 0, \ldots, \gamma_H > 0$ *the precision parameters.*

The ASWWL kernel is positive definite as a tensor product of positive defined kernels. The time complexity of ASWWL is: $O(NH\delta n + NPHn\log(n) + N^2HPQ)$ as we now need to project and sort the data for all iterations and build $H+1$ distance matrices.

The advantage of this approach is to better control the weights given at each iteration to make an automatic selection of interesting iterations. The biggest downside with this approach is the optimization of the precision parameters $\gamma_0, \ldots, \gamma_H$ that increases the cost of learning with a Gaussian process regression (or other kernel regression method) and thus limits the number of iterations to keep.

C.2 Application of uncertainty quantification: characteristic curves

We present an example of the use of GP regression to obtain prediction uncertainties for a practical computational fluid dynamics scenario.

We focus here on the dataset `Rotor37` where inputs are triplets composed of the finite element mesh of a 3D compressor blade, the (scalar) rotational speed and the (scalar) input pressure. During the design of such 3D compressor blades, characteristic curves are often used to visualize the evolution of several physical quantities in different operating regimes. Such characteristic curves allow the engineer to study the influence of geometric variations, which are relevant to determining an optimum geometric shape and operating parameters. When considering a new blade, the engineer does not necessarily know if it belongs to the training distribution, so he needs to know the uncertainties to assign a level of confidence to the prediction.

Figure 6 shows four curves corresponding to four different input rotations and twenty input pressures that go beyond the ranges of the train input pressures for a new test blade. We consider two extra (scalar) output quantities of interest, namely massflow and compression rate (not mentioned in the paper) that characterize the engine performance. Confidence intervals for the three (scalar) outputs are represented. We observe that the uncertainties are reduced near the training domain compared to the uncertainties obtained when the model is applied outside the support of the training distribution. The Gaussian process therefore gives coherent uncertainties when using the SWWL kernel.

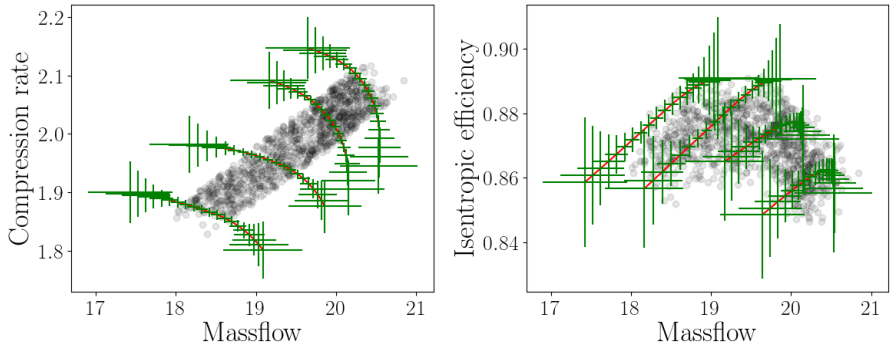

Figure 6: Graphs of the predicted compression rate and isentropic efficiency with respect to the massflow for a test mesh, 4 input rotations, and 20 values of input pressures that go beyond the range of the train and test datasets. The green lines correspond to 95% confidence intervals. Black dots correspond to train outputs.