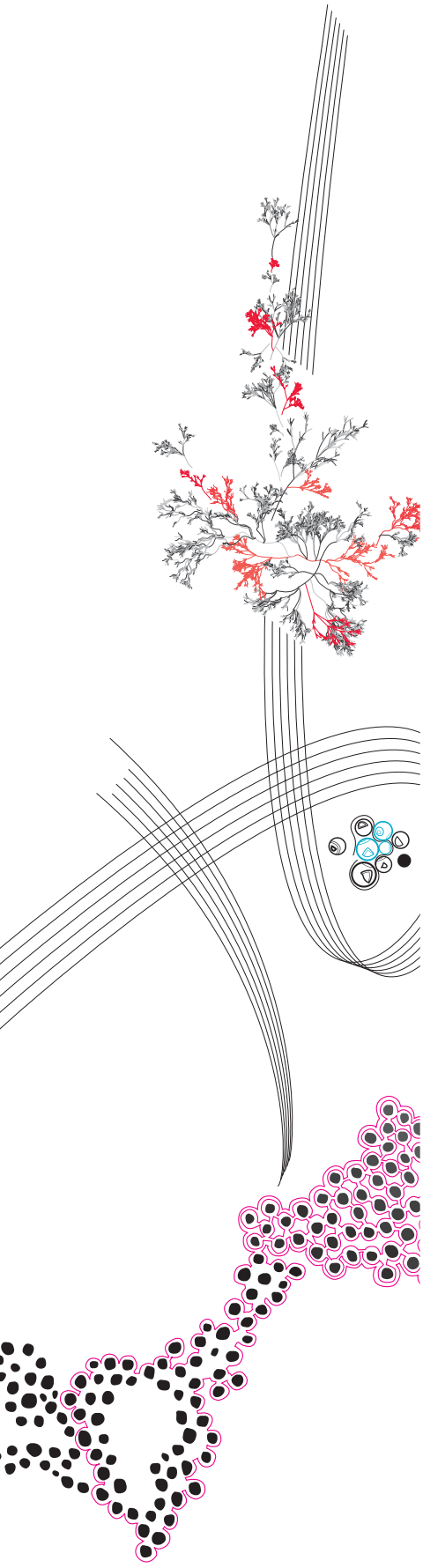


MSc Applied Mathematics
Final Project

PDE Based Neural Networks for Arterial Hemodynamics Estimation

Jente van Werven

Chair: prof. dr. Christoph Brune
Daily Supervisor: dr. Jelmer M Wolterink
Daily Supervisor: Julian M. Suk, MSc
External Member: dr. Felix L. Schwenninger

August, 2023

Mathematics of Imaging and AI (MIA)
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

CONTENTS

PREFACE

I would like to thank Julian Suk for his enthusiasm and support during this project. When I ran into a problem, you were always eager to help me, whether it was about math or other things. I really appreciate all your effort and support in helping me complete this thesis. I would also like to thank Jelmer Wolterink for supervising me, I enjoyed this project a lot. I learned a lot from our meetings, and your feedback, and I want to thank you both for encouraging me to keep improving. I would also like to thank the additional members of my graduation committee, Felix Schwenninger and Christoph Brune, for reading and evaluating my work.

Thank you to my family for helping me through the last 7 years. Thank you to my friends, who cheered me on and supported me when I needed it and helped me through difficult times. Thank you to Nienke and Lucas for your emotional support throughout this project, and to Lotte, Lucas and Jarco for proofreading my work. Thank you to the amazing and kind people I met during my time at the UT.

ABSTRACT

Learning on manifolds is an important and difficult task within deep learning. In order to learn on a manifold, it is necessary to first discretize it. Many graph and mesh based techniques overfit to the particular discretization of the discrete representation of the manifold. DiffusionNet [35] [1] shows multiple contemporary networks suffer from sensitivity to discretization, and proposes spatial diffusion to define feature communication on the manifold. This makes their network less sensitive to mesh discretization.

However, diffusion acts as a spectral low-pass filter, removing detail from the signal. Additionally, diffusion encourages local feature sharing but does not offer support for long-distance feature sharing beyond diffusing the signal to a global mean.

We investigate the use of the hyperbolic wave equation to replace the diffusion dynamics in DiffusionNet. We evaluate our method by learning biomedical signals dependent on local and global manifold structure. We utilize a spectral kernel to evolve our network layers via the wave equation, and show this kernel drops less high frequency coefficients from the signal than diffusion. Finally, we derive several network architectures based on DiffusionNet using the spectral wave kernel, and show they are outperformed by DiffusionNet.

[1] Sharp, N., Attaiki, S., Crane, K., & Ovsjanikov, M. (2022). Diffusionnet: Discretization agnostic learning on surfaces. ACM Transactions on Graphics (TOG), 41(3), 1-16

1 INTRODUCTION

Predicting and estimating signals on graphs and meshes has applications in biology [39], chemistry [17], and earthquake epicentre estimation [18]. Neural networks provide computationally fast solutions while not explicitly requiring a model of the underlying signal dynamics. A *graph neural network* (GNN) takes as input a graph or mesh. The GNN uses the topological structure (edges, neighbourhoods) of the manifold that the signal lives on to learn a set of parameters that maps to the desired output signal, via data-based training.

The *message passing* paradigm is a common structure for GNNs. In *message passing graph neural networks* (MPNN), nodes in the graph exchange information about their features. Which nodes are permitted to exchange information is controlled by how many edges lie between them. For example, a 1-hop message allows neighbours to communicate their features with each other.

This structure of communicating and updating node features via neighbourhoods works well on graphs [17]. However, because neighbourhoods on a mesh are products of the discretization of the surface rather than the surface itself, using neigbourhoods to define feature communication is inconsistent. This is because there are many different ways to discretize the same surface into a mesh. A neigbourhood on a fine mesh may enclose a smaller area than on a coarser mesh of that same surface. Moreover, for some signals adaptive meshing is used, which implies finer meshing in areas of high signal variation. Using neighbourhoods to define message passing radius could result in overfitting to the mesh discretization [39].

Therefore, we look for a way to define the radius of information sharing on the manifold rather than the mesh, minimizing the effect of the particular mesh discretization. One method recent works have attempted is using a *partial differential equation* (PDE) to determine the information sharing process on the manifold. Let Δ be the *Laplace-Beltrami operator*, the Laplace operator for a manifold. DiffusionNet [35] uses the heat equation, known as

$$\frac{\partial u}{\partial t} = \Delta u \tag{1.1}$$

to define the contribution of vertices to the feature update. DiffusionNet takes as input a signal defined on each vertex of the mesh, known as the input features. This vector is then diffused spatially via the heat operator $H(t) = e^{-\Delta t}$. The input features can be seen as the initial condition of the heat equation (1.1).

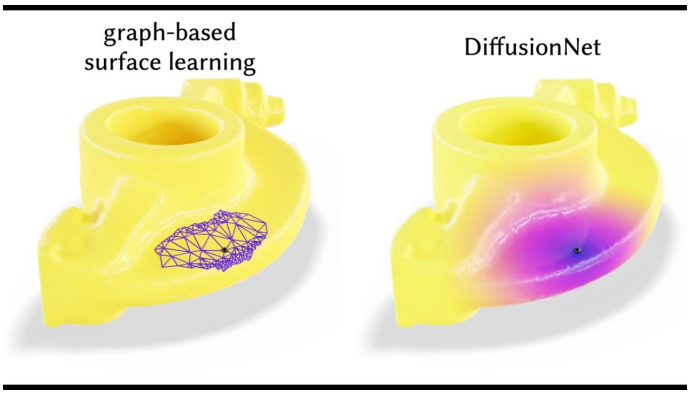


Figure 1.1: As visualized on the right, diffusion operates smoothly on the manifold, a clear advantage over the graph based method. However, further nodes have strictly less influence than closer nodes on the convolution, suggesting a potential limitation in long-distance feature communication. Image taken from [35] (twitter)

At the first layer of DiffusionNet, the input signal is diffused to a learned time t, which may vary per channel. Then the spatial gradient of this diffused signal is found by a finite difference method, and multiplied by a learned linear layer. This learned spatial gradient feature allows the network to learn anisotropic filters. Finally, the original signal, diffused signal, and spatial gradient feature signal are concatenated and passed through a *multi-layer perceptron* (MLP).

However, DiffusionNet has two shortcomings. First, the diffusion equation from a spectral perspective is a *low-pass filter*. Low-pass filters reduce the importance of high-frequency components. These high frequency components are generally understood to be responsible for defining detail in the signal. Using only diffusion to share vertex features may result in the shared signal being too robbed of detail.

Secondly, from a spatial perspective diffusion simulates short-range feature communication. Messages from distant neighbours are assigned strictly less weight than those from close neighbours, making long-range feature communication difficult.

We augment DiffusionNet with the dynamics of the *hyperbolic wave equation.* We show this PDE acts like a band-pass filter spectrally, and spatially allows for long-range feature communication. Specifically, we design three networks. The first is *WaveNet*, which structurally is identical to DiffusionNet, but uses the spectral wave equation to propagate its features to allow the network to define long-range interactions. Secondly, we use a combination of the heat and wave PDE, *WaveDiffusionNet* to first diffuse and then propagate the signal from a feature channel. Our third network, the *Parallel PDE Channels* (PPC) evolves each feature channel according to different PDEs and to different learned times, allowing the network to specify multiple short and long range interactions for a channel within in a single layer.

Several recent works on GNNs use the hyperbolic wave equation. These methods discretize the PDE via implicit, explicit, a combination (known as *ImEx*), or multi-step methods. In contrast, we apply the spectral wave kernel developed for signals on graphs by [18] to evolve the feature vector of WaveNet via the wave equation. Unlike explicit schemes, this method is stable unconditional on the timestep. Unlike implicit schemes, this method requires no large linear system to be solved.

We test our network on a set of 2,000 synthetically generated coronary artery meshes, each containing around 20,000 vertices. The mesh signals are the *wall shear stress* (WSS) and

blood pressure (BP), both shown to be biologically relevant in assessing arterial health [42]. Both the WSS and the BP are computed via *computational fluid dynamics*, a way of numerically evaluating a PDE. The WSS is influenced by local geometry, whilst the BP is caused by global dynamics within the artery. This makes them suitable to evaluate the short and long-range information sharing capacity of our network.

We show that the spectral wave kernel produces wave-like behaviours, that are dependent on the amount of basis vectors used in the projection. We additionally show a feature vector propagated by the spectral wave kernel behaves in a wave-like manner.

We show that DiffusionNet reaches near state of the art performance on the above dataset. Networks using the wave equation get outperformed by DiffusionNet.

1.1 Structure of this report

In chapter 2 we provide background on neural networks on graphs and meshes, as well as theory needed to define and solve diffusion and wave equations on surfaces. We also provide a brief introduction to spectral filtering, with focus on the intuition behind low and high frequency components within a signal. In chapter 3, we discuss the dataset, neural network architectures used, and evaluation metrics. We explain the experiments on estimating WSS and BP, and an additional experiment detailing the ability of WaveNet to estimate high frequency signals. In Chapter 4 we show our results and learned time parameters of the networks. A conclusion will be provided in Chapter 5, and we discuss our results, including the shortcomings of the use of the spectral wave kernel within our neural networks. We finally provide directions for future research.

2 BACKGROUND

2.1 Deep learning

A neural network, in its most general form, is a set of learned parameters that define a function that maps the training input to some desired output. The output is often referred to as the *label*, and can take any shape: in *classification*, it is the class number, in *segmentation* it can predict per element of the training the class it belongs to, and in *regression*, the most general problem, it predicts per element a scalar or vector. We generally represent the network as a *graph*, this allows us to visualize the way the network parameters are connected to one another.

Recent advancements have been made in image segmentation, language models etc. In all of these cases, a neural network was trained using a large amount of data to make predictions about new data.

We aim to estimate a signal on a manifold using neural networks, because evaluating a neural network is quick, and the networks can learn any signal associated to the input mesh regardless of whether we understand the underlying signal dynamics. However, defining a neural network on a mesh is not trivial, because we wish to use the underlying structure of the manifold the mesh represents.

2.2 Learning on Manifolds

Manifolds are embeddings of high dimensional spaces in low-dimensional settings. Some common manifolds are: spheres and other surfaces, the mobius strip, and R^N. Being able to learn and estimate signals living on manifolds is crucial.

One example of a complicated yet vital surface we wish to learn about are arteries. Some important signals living on these manifolds are the blood pressure (BP) at every point on the surface, as well as the wall shear stress (WSS) vector, again defined for every point on the surface. The WSS is a vector that points in the direction of the stress force acting on an artery. Both the BP and WSS are related to arterial health [42].

Ultimately, we cannot learn directly on the continuous manifold of, for example, the blood vessel. This is due to computational limits and the fact that we cannot form a continuous model in a computer simulation. Instead, the surface needs to be discretized to learn the signal. We focus specifically on methods that discretize the manifold the data lies on into either graphs or meshes. In our overview of current methods, we follow [7].

We first introduce prominent techniques for learning on graphs. This is followed by an example of architectures that work well on graphs not being robust to remeshing. We then introduce methods specific to meshes, including the key work inspiring our own investigation, DiffusionNet

[35].

2.2.1 Graph neural networks

In a graph neural network (GNN), an input graph containing features defined on nodes and edges is passed through several network layers. For example, for predicting whether new edges will form between users of a social media platform, the nodes are the users, the edges whether they are friends on the platform, and the node features could include their hobbies, age, or geographical location. The GNNs task is then to use the information on the graph to predict where new edges will develop between users.

Message passing neural network [17] The message passing paradigm is a strategy designed for graph neural networks. In each layer of an MPNN, nodes may look at surrounding nodes within a number of edge hops of them. They aggregate the features using an aggregation operator. This operator must be permutation-invariant, so that the order in which the node features are seen does not matter. This is because there is no ordering consistent way to order the neighbours. The node then updates its own features by passing the aggregated information from its neighbours through a shared non-linear layer (MLP).

The most general form of GNN, the Message Passing Neural Network (MPNN), where the node hidden state h_v^t is updated via

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{2.1}$$

$$m_v^{t+1} = \left(\sum_{j \in \mathcal{N}_v} M_t(h_v^t, h_j^t) \right) \tag{2.2}$$

where the $\sum_{j \in \mathcal{N}_i}$ can be replaced by any permutation-invariant operator [17]. In the equation above, U_t is some vertex update function shared for all vertices, m^{t+1} contains the aggregated messages from the surrounding nodes, and M_t is a function that computes the message. The two other major paradigms, Attentional GNNs and Convolutional GNNs can be expressed as specific instances of MPNN.

Attentional GNN: rather than expressing fixed or learned weights relating features from one node to another, Graph Attention Networks [43] define the weights to be a learned linear function dependent on features of both nodes. Let h_i denote the value of node i, and α_{ij} the edge weight between node i and node j. Then the attention-based update of the feature h_i is given by

$$h_i = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} g_j \right), \; \alpha_{ij} = a(h_i, h_j) \tag{2.3}$$

Convolutional GNN: we can leverage the spectral convolution theorem to define node interactions. This style of aggregation leads to fixed edge weights based on graph connectivity. [6] As explained in section 2.10, a spectral transform may be defined for graphs, where V denotes the spectral basis and Λ a matrix of eigenvalues. Then we may convolve a filter $g(\cdot)$ with our signal u as

$$(g * u) = V^T g(\Lambda) V u \tag{2.4}$$

Such networks were first popularized by [9]. Notably, the derived eigenbasis is dependent on the mesh, and learned filters $g(\cdot)$ often do not represent the same information within different

meshes [7]. For this reason, [12] uses properties of the graph wavelet transform [20], specifically the result relating the localization of a filter to the polynomial of its coefficients, to define filters that are localized to $K-hops$ of their central node. Recently, several works utilize wavelet bases on graphs, inspired by the spectral graph wavelet transform designed by [20]. For a more complete overview, we refer the reader to [7]

PDE Based: this is a loosely defined form of GNN, where the node dynamics are enforced via PDEs rather than learned spectral filters or attention mechanisms. PDE based methods provide both the message passing and vertex update rules for MPNN. However, it is argued in [6] that we should see these networks as separate from MPNN. These networks provide interpretable dynamics that evolve the features. Additionally, these dynamics do not require 1-hop or 2-hop convolution sizes, rather, they are defined on the manifold [6]. There are several ways to define such a network, in general some non-discretized PDE operator L functions as the network propagator:

$$h_t = L(u, t) \tag{2.5}$$

for some input feature u and hidden layer h_t. PDEs are processes linking time and spatial domains, and therefore a natural generalization of GCNs can be made by interpreting them within this framework.

In [14], the authors build upon previous work [33] that explored the connection between ResNet [21] CNNs on a grid and PDEs. They motivate searching for this connection by explaining PDEs are well-understood and implementing networks via PDEs can have computational advantages. Their work [14] proposes using a forward time discretization of a hyperbolic non-linear PDE and parabolic non-linear heat equation to define network layers. Specifically, they solve the non-linear heat equation using forward Euler and the non-linear hyperbolic wave equation using the leapfrog method. Their network, PDE-GCN, uses a learned convex combination of both PDEs to model its dynamics.

In [32], the hyperbolic wave equation is used to create a graph neural network that preserves Dirichlet Energy, combating oversmoothing. Additionally, they reformulate a general GNN as a PDE and subsequently observe that common MPNN and attentional network structures can be related to steady states of the PDE defined by the features and layers. Let $F(\cdot)$ denote the network parameters, and X^n the feature vector at layer n. Then

$$X^n = \sigma F(X^{n-1}) \tag{2.6}$$

By instead formulating their GNN via the discretized of the full PDE rather than the steady state, the authors expect to gain access to a wider range of dynamics.

Another work related to PDEs on graphs is [2], which builds on DiffusionNet's diffusive dynamics and adapts them to a graph, where the discretization is a feature rather than a limitation. In [2] the derivative of the heat kernel is used to achieve long-distance message passing on graphs.

Oversmoothing: When many layers of a GCN are stacked, the convolutional layers can lose expressiveness, because each node in the graph carries the same feature. This phenomenon, dubbed as "oversmoothing", is often proportional to the number of graph layers, and has been linked to the spectrum of the Laplace operator. [27]

To measure oversmoothing, the Dirichlet energy has been proposed by [32]. The Mean Average Distance (MAD) may also be used, although this may be computationally unstable and is

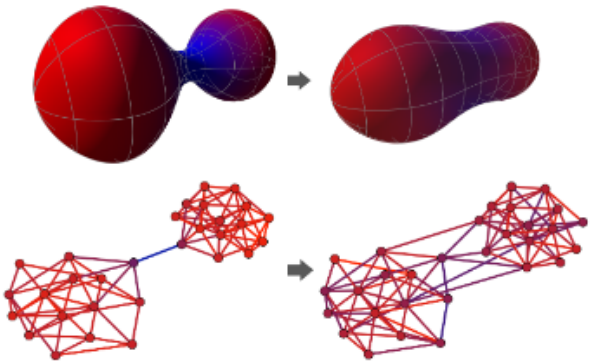


Figure 2.1: Visualization of curvature on manifolds and graphs, before and after edge replacement to reduce negative curvature. Blue/red shows negative/positive quadrature. Image taken from [41]

recommended against [31]. A comprehensive survey on the current state of oversmoothing can be found in [31]

Solutions to the problem include different forms of normalization, as outlined in [27] and [31]. Additionally, ResNet structures can help. Finally, different forms of message passing that resemble continuous-time evolution of features according to dynamical systems have recently become popular [31]

Bottleneck & Over-squashing: In some cases, such as regular message passing in a small-world graph as highlighted in [41], the receptive field of a node i grows exponentially with the radius of the convolution. This node receives messages from a large amount of nodes, that need to stay in a vector of fixed size. This results in an over-squashing of features, and thus information, in that node, making it a bottleneck. Recently, [41] has shown that edges with high negative curvature cause bottlenecks in the graph, leading to over-squashing. They introduce a new form of Ricci curvature for graphs to measure bottlenecks.

As a follow up, recent work has shown theoretically and empirically that in MPNNs over-squashing is influenced by the specific structure of the message passing operator, and its effect can be reduced by increasing the size of the hidden layers of the network. Additionally, the authors use the idea of commute times to reason about which nodes in a graph may struggle to share their information, and present graph rewiring, i.e. adding or removing edges from the graph, as a potential solution. [13]

2.3 Graph-Based Methods on Arterial Manifold

GEM-CNN [19] was recently used in [39] to estimate the BP and WSS on a synthetic arterial dataset. GEM-CNN is a mesh neural network borrowing heavily from message-passing neural networks (MPNNs), a specific form of graph neural network (GNN).

GEM-CNN uses equivariance to the rotational group $SE2$ to make its message-passing operation on the mesh equivariant. An object f is equivariant to actions of a members of a group G if $f(g) =$. We need equivariance to $SE2$ because on a mesh, vertices are connected via edges that are based on the geometric structure of the underlying manifold. In order to have access to

this information, the network needs to be able to determine the angle between nodes. Because there is no direction to call the origin, GEM-CNN takes arbitrary origins for the message-passing step and ensures the output is equivariant to them.

However, GEM-CNN is sensitive to the specific meshing used to train the network. This can be explained by the message passing paradigm intrinsic reliance on mesh structure to define which nodes are permitted to exchange features. The MPNN does so via neighbourhoods, which on a mesh can encompass very different area sizes, especially on a mesh with adaptive meshing, where specific regions in the mesh are better approximated using more triangles. The WSS and BP are properties of the underlying manifold and not its discretization, thus despite GEM-CNNs performance qualities we look for a network less sensitive to the mesh discretization.

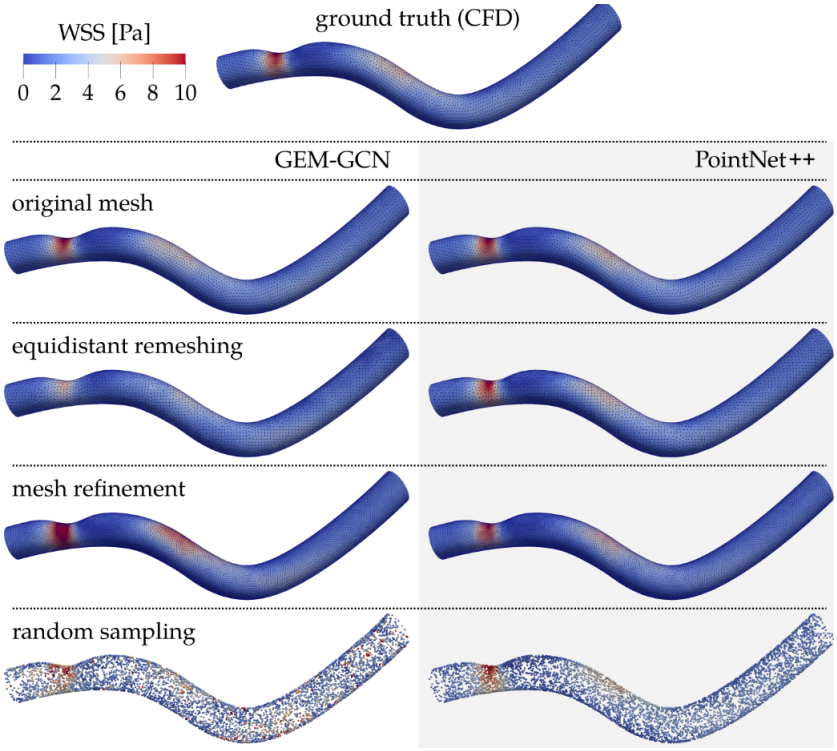


Figure 2.2: GEM-CNN (left) and PointNet++ (right) evaluated on several remeshed artery walls. GEM-CNN is sensitive to local mesh discretization. Figure taken from [39].

Ultimately, GEM-CNN is evidence that principles from GNNs do not always directly translate to meshes. GNNs exploit the structure and connectivity of their graph rather than the underlying manifold. Therefore, we investigate some networks specifically designed to estimate properties of meshes.

2.4 Mesh based methods

Most generally, mesh based methods generalize the notion of convolution to the mesh. They typically do this via a spatial perspective, using a patching operator to define their filter support.

A mesh defines a domain similar to a graph, where vertices contain features and are connected via edges. However, the mesh is also a spatial structure, and we can define the distance between two vertices in a Euclidean way. Additionally, the connectivity of a mesh is geometrically

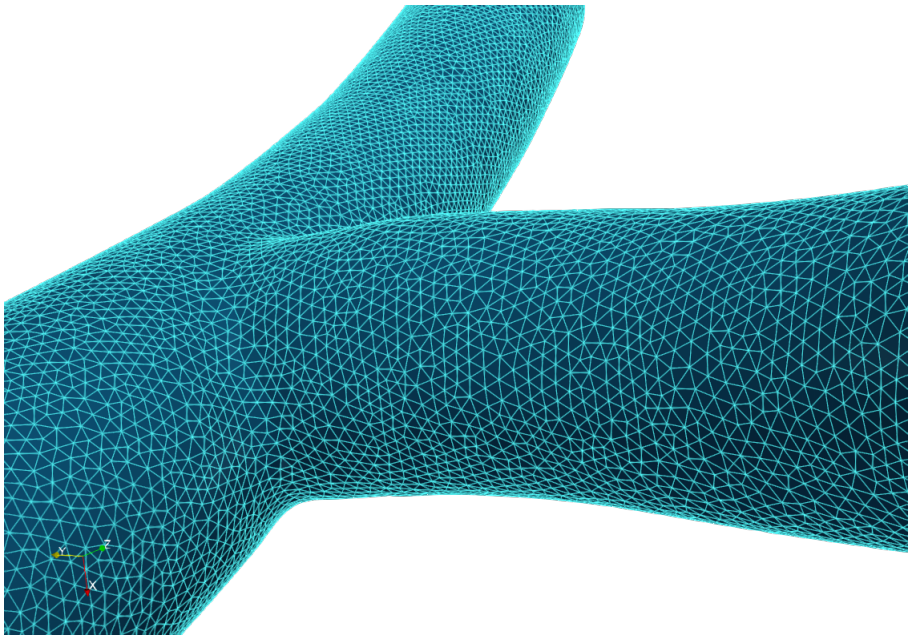


Figure 2.3: Image of a mesh of a bifurcating artery.

defined, and only vertices that are close to each other typically form faces. Thus we have a lot more structure to exploit in a mesh than a graph. We define a mesh as follows, taking inspiration from [37]

Definition 2.4.1 (Mesh)**.** A *mesh* is defined by $\mathcal{M} = (V, F)$, where $V = \{v_0, \ldots, v_n | v_i \in R^3\}$ is the set of vertices in R^3 and $F \subseteq \{1, \ldots, n\}^3$ the set of faces. Faces define connections between the vertices in V. For a triangle mesh, any $f = (f_1, f_2, f_3) \in F$ may contain at most three vertices. Additionally, the mesh must satisfy the manifold criteria: an edge touches two faces, a vertex must be surrounded by plane of faces, and that the mesh is oriented.

However, we still have no canonical ordering of the vertices. Suppose we desire to update any given node on the mesh using a weighted average of only its neighbours, whilst taking into account the angles between the vertices. Then there is no way to define which direction is up a or down, and therefore which vertex feature should receive which angle-based weight. [8]

This is an issue that has been tackled in many ways. GEM-CNN [38] defines gauge equivariant convolution kernels. By projecting the features into the local tangent space of the node being updated, and utilizing parallel transport to ensure that whichever arbitrary node we use to define our origin, we still get a similar result, gauge-equivariance is achieved.

CNN: Several attempts have been made at translating CNN architectures to irregular domains such as graphs and meshes. Geodesic CNN (GCNN) [24] uses geodesic distances to define patches on the mesh, and uses a form of template matching to construct convolutional filters for these patches. However, a limitation of this approach is that using the geodesic radius to define the patch does not always result in ”topological disks” when the radius is large. [5] Building on this, Anisotropic CNN [5] uses anisotropic diffusion to define directional patches. In anisotropic diffusion, a tensor $D(x)$ weights the diffusion depending on the direction and position

$$\frac{\partial u(x,t)}{\partial t} = \nabla \cdot D(x) \nabla u(x,t) \tag{2.7}$$

The anisotropic diffusion equation is a generalized version of the diffusion equation, and if

$D(x) = I$ we obtain the more conventional

$$\frac{\partial u}{\partial t} = \Delta u \tag{2.8}$$

The heat kernel, intrinsic to the shape, does not suffer from the aforementioned issues of GCNN as it is defined everywhere on the mesh. The anisotropic diffusion equation is incorporated into the discretization strategy for the mesh Laplacian, and a spectral solution is used for fast computation. MoNet [26] proposes a more general definition of a patching operator. Their particular experiment uses Gaussian Mixture Models with learnable mean and covariance. The authors note that Anisotropic CNN and Geodesic CNN are examples of MoNet with specific, handcrafted weight functions for the patch operator. MGCNN [30] defines equivariant convolution on meshes using directional functions and template matching.

Random Walk: Additionally, some neural networks have explored a probabilistic way of exploring the mesh: rather than sharing spatial features via convolution or message passing, MeshWalker [23] define a random walk over the mesh: at a vertex v, the walk can go to any adjacent vertex $u \in \mathcal{N}_v$ with some probability $p(v, u)$. This process is repeated a number of times to simulate walking over the mesh, hence the name random walk. The sequence of vertices is then processed by a Recurrent Neural Network (RNN), a type of network that excels in analyzing sequential data.

Recently, a PDE-based mesh neural network based on the evolution of features on the surface has emerged, called DiffusionNet [35]. This network aism to minimize the effect of mesh discretization.

2.5 Diffusion Net

In the paper on DiffusionNet [35], the authors show that a number of contemporary mesh neural networks are sensitive to discretization. Their method is built around avoiding this, by using diffusion to define their feature update.

Rather than selecting a neighbourhood of nodes that may communicate, DiffusionNet defines a spatial support for each feature channel on the mesh by learning a learned time parameter t. It then diffuses those feature channels to a time t and applies a per-vertex MLP to the feature vector. This is repeated for several layers. Because diffusion is anisotropic, spatial gradient features are used to introduce anisotropy [35].

DiffusionNet uses the heat equation on the surface of the mesh to evolve the feature vector u via

$$\frac{\partial u}{\partial t} = \Delta u \tag{2.9}$$

DiffusionNet is that the diffusion occurs in a spatially motivated way, despite having a spectral implementation. Additionally, the PDE dynamics evolve features continuously on the manifold, rather than discretely per node. This eliminates the specific local discretization of the mesh as much as possible.

By using the spectral theorem and previously discussed self-adjointness of the heat equation, we may evolve a feature vector u_0 to a diffused state u_t via

$$u_t = V^T g(\lambda) V u_0 \tag{2.10}$$

The use of the spectral heat kernel $g(\lambda, t) = e^{-\lambda t}$ is intended to allow the network to select the receptive field of the convolution. If t is large then the information sharing between vertices of the mesh is global, if t is small the reverse is true. The time t is defined per channel, and each channel may subsequently learn a different time. Below is a visualization of DiffusionNet's structure:

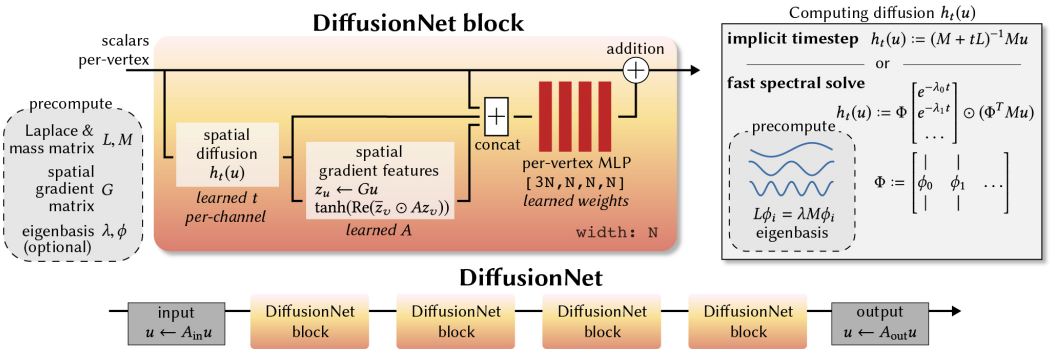


Figure 2.4: The above illustration, taken from [35], shows the structure of the layers within Diffusion Net. DiffusionNet is composed of multiple Diffusion Blocks. Inside the DiffusionNet Blocks, the input feature vector is fed to the output via skip-connection, resembling the ResNet structure. Additionally, the input feature u is evolved by the heat operator to a time t learned per-channel by each Block and concatenated to the skip-connection. Finally, the evolved signal is used to construct spatial gradient features. These three components thus form a vector of $[3N, f]$ with N the number of vertices and f the dimension of the vertex features. The per-vertex MLP is then applied independently to each node of the mesh, and the result of the MLP is added to the input signal u

DiffusionNet uses the ResNet [21] structure: given the initial feature u we learn a function residual function $F(u) = H(u) - u$ and output

$$u_{\text{out}} = F(u_{\text{in}}) + u_{\text{in}} \tag{2.11}$$

We thus indirectly learn the mapping $F(\cdot)$ and form a skip-connection from u_{in} to u_{out}. The ResNet [21] structure of Diffusion Net means layers are free to resort to the identity mapping, which makes the network less likely to overfit.

Diffusion net is composed of multiple diffusion blocks, where spatial diffusion occurs on the signal via spectral acceleration, and learned spatial gradient features create anisotropic filters. Finally, at the end of each layer a per-vertex MLP is applied to the signal to update the hidden signal, and at the final layer to predict the output signal.

2.6 Shortcomings of DiffusionNet

DiffusionNet's spatial diffusion smoothes out high frequency components of the signal that passes through its layers. This is due to the nature of the heat equation, which in the spectral domain is much like a low-pass filter.

Specifically, suppose we have some output at layer 2 of the layer, u_2. Then this signal is spatially diffused, updating each node feature by averaging the signal around it. However, spectrally we know that detailed, complicated information resides in the higher frequency eigenvectors. Diffusion lowers the coefficients of those eigenvectors and therefore spatial diffusion makes the

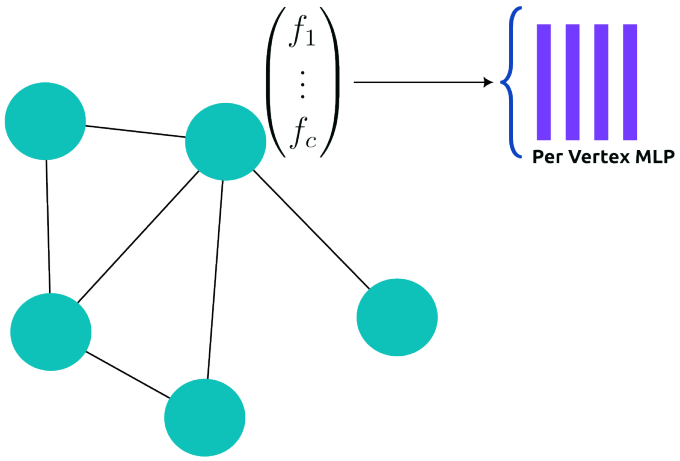


Figure 2.5: In the above visualization, each vertex has an associated feature vector $(f_1, ..., f_c)$, and a per vertex MLP is applied to this vertex's feature vector. The weights of this MLP are shared among all vertices. This also highlights the importance of message passing in the GNN: without message passing via diffusion, the MLP would have no notion of spatial structure of the mesh. [35]

signal more boring and less spatially variant. As we saw in section 3.2, the WSS signal is highly spatially variant and its spectrum contains coefficients of high-frequency eigenvectors. Thus, a low-pass filter may not provide sufficient ways to reconstruct such a signal.

Secondly, from a spatial point of view, information cannot be exchanged between two nodes without also including the nodes between them. We can see this visually in Figure 2.7. In this Figure we show a point source diffused to two different times. The degree to which the center node affects those around it is directly proportional to the distance.

Within the context of DiffusionNet, in the hidden layer update the vertex features at the point source would be updated via a weighted average of all vertices on the mesh, with the weighting defined by how red the color of the vertex is. We see that close vertices always get higher weights than distant vertices. DiffusionNet is only able to let distant vertices' features affect each other if the signal is diffused for a very long time, smoothing out any detail within it. This lack of long-range feature communication could affect DiffusionNets ability to reconstruct complicated signals on a mesh.

The authors mention in their work [35] that they acknowledge this smoothing out of features, but believe it is sufficiently compensated for by the MLPs in the layers.

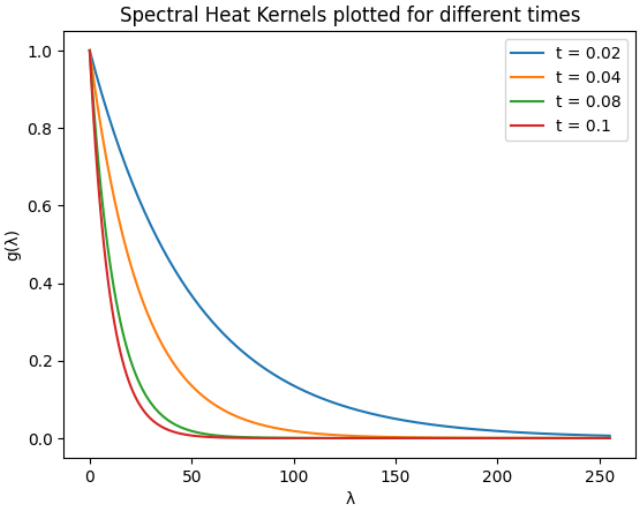
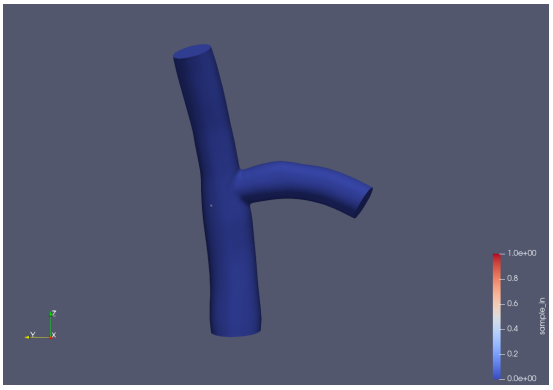
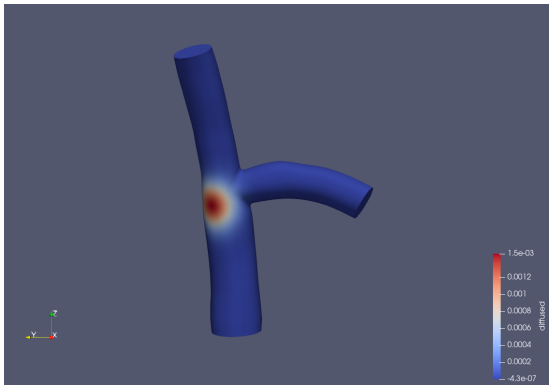


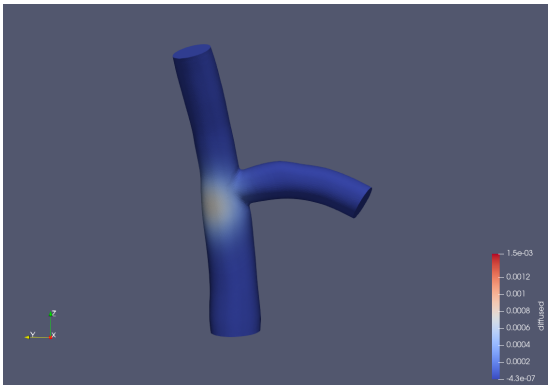
Figure 2.6: The heat kernel $g(\lambda) = e^{-\lambda t}$ is not capable of expressing preference to high frequency features.



(a) Location of point source (red dot)



(b) Point source at t_1 under diffusion dynamics



(c) Point source at $t_2 = 2t_1$ under diffusion dynamics

Figure 2.7: A point source is propagated via the heat kernel using a reduced basis of 256 eigenvectors. The diffusion kernel is not capable of expressing that a vertex further away should be equally or more important than a vertex close by.

2.7 PDEs on surfaces

In order to better understand what it means to model our neural network dynamics based on PDEs, we discuss the heat and wave equations on a surface. We start with some definitions:

Definition 2.7.1 (Laplacian)**.** Let $V \subseteq R^n$. The Laplacian Δ of a scalar function $u : V \to R$ is given by $\Delta = -\text{div}(\nabla u)$, i.e. the divergence of the gradient of u

Since we are on a Riemannian manifold, we instead have the Laplace Beltrami operator. This operator simplifies to the Laplacian on R^N with Cartesian coordinates, and adapts the notions of divergence of the gradient to a Riemannian manifold.

We can interpret the gradient ∇u as a way to describe the direction the function u is going. The divergence at a point $p \in V$ can be interpreted as a measure of how much of the vector field is leaving versus entering an infinitesimal region dV. The divergence of the gradient, Δ, represents the deviation from the value of p in a neighbourhood around p.

We could model the heat distribution of a room by, for example

$$\frac{\partial u}{\partial t} = c\Delta u + v(p) \tag{2.12}$$

where $v(p)$ is some potential function, possibly dependent on the spatial variable p, and c is a diffusion coefficient, which we assume to be constant in both time and space. The equation models the heat distribution over time, relating the time evolution of the temperature $\frac{\partial u(p)}{\partial t}$ at a point p to the average temperature around it, $\Delta u(p)$.

The boundary conditions we impose on our PDE cause it to have a unique solution, if one exists. Another common PDE is the wave equation, known for its propagational dynamics. Many different versions of wave equation exist, we choose to focus on the two-way wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2\Delta u \tag{2.13}$$

We can group second order PDEs by the behaviour they commonly exhibit as a result of their coefficients. Three groups exist: parabolic, hyperbolic, and elliptic PDEs. The heat equation is an example of a parabolic PDE, while the wave equation is a hyperbolic PDE. An elliptic PDE is characterized by the smoothness of the solutions, and often the level sets of parabolic or hyperbolic PDEs result in elliptic PDEs. An example of an elliptic PDE is the Laplace equation $\Delta u = 0$.

2.8 Self-Adjoint Operators

Naturally, we desire a method of determining the solution to the PDEs that define our network dynamics. To do this, we reformulate our problem in terms of functional analysis, and use the spectral theorem to reformulate the PDE into a simpler form.

Definition 2.8.1 (Adjoint Operator)**.** Let $\langle , \rangle$ denote the inner product on the finite-dimensional Hilbert space V, and let L denote a not necessarily bounded operator with domain V. Then the adjoint operator denoted by L^* is the operator that satisfies $\langle Lx, y\rangle = \langle x, L^*y\rangle$

Definition 2.8.2 (Self-adjoint Operator)**.** An operator L is said to be self-adjoint if it is its own adjoint, i.e. $L^* = L$

A self-adjoint operator therefore commutes within the inner product

$$\langle Lf, g\rangle = \langle f, Lg\rangle \tag{2.14}$$

One property we can learn from this is that any eigenvalue must be real. Following [4], we let v be an eigenvector of L with eigenvalue λ and unit length. Then

$$\begin{aligned}\lambda &= \langle Lv, v\rangle \\ &= \langle v, Lv\rangle \\ &= \overline{\langle Lv, v\rangle} \\ &= \langle v, Lv\rangle \\ &= \overline{\lambda}\end{aligned}$$

and the only λ that satisfy $\lambda = \overline{\lambda}$ are $\lambda \in \mathbb{R}$. Crucially, if $\mathcal{L}$ is self-adjoint, it can be diagonalized into

$$L = V\Lambda V^T \tag{2.15}$$

where Λ is the spectrum of $\mathcal{L}$ and V the orthogonal stacked matrix of eigenfunctions. This is especially important in making sense of the operator exponential e^L. The operator exponential occurs naturally in semi-group theory, a branch of mathematics that can be used to organize all solution operators of a PDE by defining the infinitesimal generator of a semi-group. More importantly, we can use this to solve evolution equations. Many PDEs may take the form of

$$u(t) = e^L t u_0 \tag{2.16}$$

When L is self-adjoint, we can diagonalize this expression using the spectral theorem, providing us with a natural way of analyzing a certain class of PDE operators. Not all PDE operators are self-adjoint, but a few important ones, such as the heat operator, and the Laplace operator, are.

2.9 Discretizing the PDE

Because we cannot directly work with the continuous manifold our signal is defined on, we instead study the PDE on graphs and meshes. We first need a discrete approximation of the Laplace-Beltrami operator. This discretization is not an artefact of FEM, rather, many different mathematical approaches (FEM, discrete exterior calculus) arrive at the same discretization [46]. The derivation below is based on FEM and follows [11][36]. Let us consider the Helmholtz equation that defines the eigenfunctions of the Laplacian, with Dirichlet boundary conditions

$$\Delta u = \lambda u \tag{2.17}$$

$$u|_{\partial\Omega} = 0 \tag{2.18}$$

This is known as the strong formulation of the PDE. The strong form is difficult to solve directly, so instead we solve the weak form, that gives us access to a class of solutions with less restrictions on them. To obtain the weak form, we multiply with a test function $\phi \in H^2$ integrate both sides

$$\int_\Omega \phi\Delta u = \int_\Omega \phi\lambda u \tag{2.19}$$

Applying Green's First Identity, otherwise known as integration by parts, we obtain

$$\int_\Omega \nabla\phi \cdot \nabla u - \int_{\partial\Omega} u\nabla\phi = \int_\Omega \phi\lambda u \tag{2.20}$$

By using the boundary conditions, the integral over the boundary $\partial\Omega$ vanishes and we obtain

$$\int_\Omega \nabla\phi \cdot \nabla u = \int_\Omega \lambda\phi u \tag{2.21}$$

Via the Galerkin Approximation Theorem, we may approximate the solution to this PDE on our mesh via the use of piece-wise linear basis functions. These functions, commonly known as hat functions, can be differentiated to obtain a solution to the PDE. The right hand side can conveniently be arranged into the cotangent matrix, whilst for the left hand side we are left with a product of basis functions, which can be structured into what's known within FEM as a mass matrix. The discretized Helmholtz equation is then

$$Lu = \lambda Mu \tag{2.22}$$

Unfortunately, the mass matrix is not diagonal. It has off-diagonal entries in a similar structure to the adjacency matrix. When we use the formulation as derived from FEM, the mass matrix is called consistent, and when we choose to approximate the mass matrix as diagonal, it is called the lumped mass matrix. [36]
The matrix L is known as the cotangent matrix, first introduced by [29]. Many variations of the discretized Laplace operator on a mesh exist, each with their own avdantages and disadvantages. For example, the cotangent Laplacian is worse for diffusion purposes because the edge weights are allowed to be negative for non-manifold meshes [48]. For a thorough treatment of different discretizations and their strengths we recommend [48].

2.10 Solution Methods

Explicit Solution: based purely on the taylor expansion of the PDE around a point, both its numerical stability and accuracy decrease as t gets larger [49]. However, the method only requires a forward pass, through matrix multiplication, making it faster than the aforementioned implicit solution at the expense of accuracy

$$y_{n+1} = y_n + hf(y_n, t_n) \tag{2.23}$$

Implicit solution: follows a similar procedure, but instead formulates the solution in an implicit way, often requiring us to iteratively find the solution using for example a root finding method [49].

$$y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \tag{2.24}$$

In the implicit solution of the PDE, we discretize the time using the implicit scheme for ODE's, while discretizing in space using the cotangent Laplacian. The resulting matrix system is positive semi-definite (PDS), and therefore allows a Cholesky decomposition. This means it has a faster solution than a general system of equations. Additionally, the implicit method is stable for large time steps. However, solving with the cholesky factorization can take a lot of space on the GPU. Additionally, this evaluation method of the PDE requires solving a system of equations for each time t, and we solve a different PDE for each channel. Both of these issues make it undesirable for use within a neural network.

Implicit-Explicit and Multi-Step Schemes Implicit-Explicit (ImEx) schemes are methods of discretizing the PDE that borrow from both implicit and explicit methods. These methods attempt to combine the unconditional stability of the implicit method with the ease of evaluation (not

needing to solve a linear system) of the explicit method. For example, [32] uses the symplectic Euler method to solve the wave equation with a damping term. Symplectic Euler discretizes a pair of coupled ODEs by using the implicit method for one of them, and the explicit method for the other, resulting in better stability. Similarly, [14] uses the leapfrog method to discretize the wave equation describing their network dynamics.

Spectral Solution If L is self-adjoint, then by the spectral theorem $L = V^T \Lambda V$. Therefore, discretizing the spatial Laplacian amounts to solving an eigenvalue problem. When the domain does not change, this eigenbasis can be conveniently precomputed and the solution formulated as an evolution equation. Suppose some PDE is solved by the operator e^A, then

$$u(t) = e^{At} = e^{V^T \Lambda V} u_0 \tag{2.25}$$

A matrix M is Hermitian, or symmetric if A is real, if $A^T = A$. A Hermitian matrix is a self-adjoint linear operator, and admits a spectral decomposition. Computing the matrix exponential for a general matrix is an expensive procedure, however for a Hermitian matrix this is simplified into

$$e^{At} = e^{V^T \Lambda V t} = V^T e^{\Lambda t} V \tag{2.26}$$

where $V = V^T$ is the unitary matrix of stacked eigenvectors of A and Λ the matrix of corresponding eigenvalues. In the above equation, $e^{V^T \Lambda V t} = V^T e^{\Lambda t} V$ due to $V = V^{-1}$. Evaluating $B = e^{\Lambda t}$ amounts to elementwise exponentiation via $B(i,i) = e^{\Lambda(i,i)t}$ due to the diagonal nature of Λ. Generally, if the differential operator is self-adjoint, the discretization attempts to preserve this property [48] [46]. This is true for the Laplace operator, and the heat operator $H(t) = e^{-\Delta t} H(0)$:

$$u_t = e^{-\Delta t} u_0 = e^{V^T - \Lambda V t} u_0 = V^T e^{-\Lambda t} V u_0 \tag{2.27}$$

Thus, a Hermitian matrix may be decomposed into its eigenbasis, and its exponentiation is a quick operation. This provides a computationally efficient way to map an initial state u_0 to a state u_t, at the expense of precomputing the eigenbasis once (since the eigenbasis is intrinsic to the mesh). For deep learning, such a fast way to evolve an initial state to a given time t is essential.

2.11 Fourier Basis

The Spectral Theorem plays a key role in evolving certain PDEs. The Fourier transform is intimately linked with the Spectral Theorem. By extent, we can use Fourier theory to investigate how our signal is being manipulated by, for example, the PDE dynamics. In the Fourier perspective, we interpret the eigenbasis of the Laplace operator as a Fourier basis. Before we discuss the Fourier transform on meshes, we first give some intuition on how it can be used to manipulate functions living on images.

Loosely speaking, we can define the Fourier transform for a function of two spatial variables as follows [25]:

$$F(\omega_1, \omega_2) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f(m,n) e^{i\omega_1 m} e^{i\omega_2 n} \tag{2.28}$$

We can interpret this as projecting our signal onto a basis of sines and cosines of varying frequency ω. Crucially, these basis functions are mutually orthogonal, facilitating projection onto them. Let $n \neq m$ be integers, then with the standard inner product $\langle f, g \rangle = \int_{-\infty}^{\infty} f(x) g(x) dx$

$$\langle \sin(nx), \sin(mx) \rangle = \int_{-\infty}^{\infty} \sin(nx) \sin(mx) dx = 0 \tag{2.29}$$

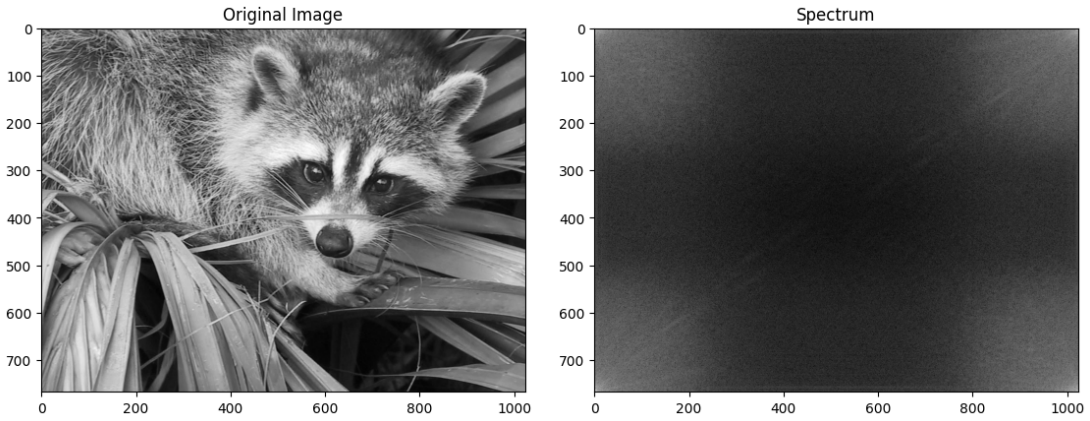
This is known as the *orthogonality of sines*. Further, they indeed form a basis, via the Spectral Theorem and the self-adjointness of the Laplace operator.

For simplicity's sake, let us consider only images in greyscale, i.e. without colours. In the discrete image domain, an image is known as a function of its pixels, let $I = (0, ..., M) \times (0, ..., N)$, then $f(m, n) : (m, n) \to [0, 1]$ for some $(m, n) \in I$, where $f(m, n)$ expresses the grayscale intensity of pixel (m, n). Then the Fourier transform can be expressed as [25]

$$F(p, q) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(m, n) e^{-\frac{i2\pi pm}{M}} e^{-\frac{i2\pi qn}{N}} \tag{2.30}$$

The resulting function $F(p, q)$ is again a function defined on I. Typically, the low frequency components, i.e. (p, q) close to the boundary of I, contain the most information, and the high frequent components generally contain sharp edges, details, and noise.

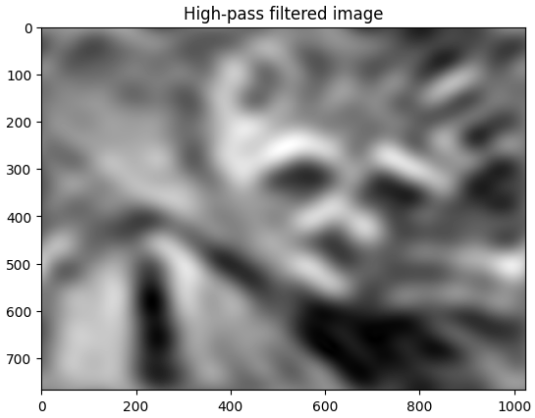
When we wish to manipulate the image, we can use our knowledge of where the information resides to attenuate specific frequencies by designing a function that modulates (p, q). For example, a low pass filter is designed to lower the weighting of high frequency components. Because noise often resides in the higher end of the frequency, this filter is effective at denoising. Similarly, a band-pass filter lowers the weighting of anything outside its particular frequency band.



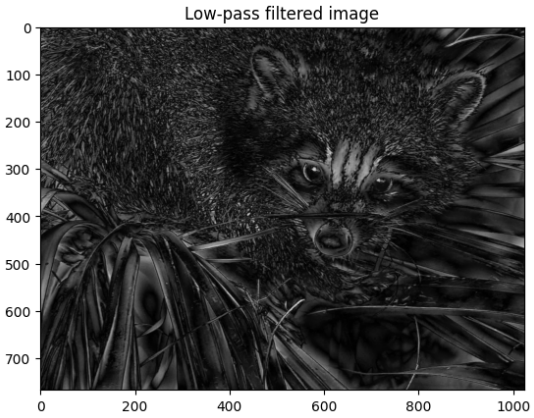
(a) Original raccoon image (b) Spectrum of raccoon image

Figure 2.8: Python implementation adapted from scipy-lectures

Consider the image of the racoon above. We may take the Fourier Transform of it, resulting in 2.8b. This frequency spectrum may be modulated as we see fit. In our case, we wish to remove low frequency components, so we set the coefficients of the first 10 components in the x and y direction to zero. After applying the inverse Fourier transform we obtain the images below.



(a) Low-pass filter (keep first 10 components)



(b) High-pass filter (drop first 10 components)

This is known as a high-pass filter, and the result is shown in 2.9b. We see that sharp edges and other details are the only information preserved. We repeat this procedure but instead build a low-pass filter that keeps only low frequency components. The resulting filtered image 2.9a is blurry, almost all detail has been removed.

By filtering, we modulate specific basis functions corresponding to certain image frequencies. By understanding the structure of the eigenbasis of the Fourier operator, we can reason about which components of the signal we keep or discard.

The basis the Laplace operator defines is closely related to the Fourier basis for R^N. In fact, the Laplace operator defines a generalized Fourier basis for the manifold, in our case a mesh. Therefore, many concepts can be transferred from traditional signal processing to our current problem.

2.12 Fourier on Meshes

For a mesh of N vertices we define our Fourier basis $V = [\phi_0, ..., \phi_N]$ such that

$$\Delta = V^T \Lambda V \tag{2.31}$$

via the spectral theorem. The *eigenfunctions*, or in the mesh domain *eigenvectors* by virtue of their discretized nature, can be seen below. Note that with increasing λ we see an increase in spatial variation of the eigenvector. This can be viewed as intrinsic to the Laplace operator: the lowest eigenfunctions are the smoothest, where the Laplace operator is minimized.

Let f be a function defined on the vertices of a graph of N nodes, $f \in R^N$. The analogue of the Fourier transform for meshes and graphs is given by [20]

$$\hat{f}(l) = \langle \phi_l^T, f \rangle = \sum_{n=1}^{N} \phi(n) f(n) \tag{2.32}$$

In other words, to find the frequency response at frequency l of our signal f we project f onto the basis vector associated with λ. The inverse transform is similarly defined

$$\hat{f}(l) = \langle \phi_l, f \rangle = \sum_{n=1}^{N} \phi(n) f(n) \tag{2.33}$$

In matrix notation, a signal may be projected into the spectral domain and back via

$$f = V^T V f \tag{2.34}$$

An important caveat is that the matrix V is only unitary, i.e. $V^T = V^{-1}$ because the discretized Δ is symmetric. For directed graphs (digraphs) or graphs with disconnected components this is no longer the case, and we must actually compute V^{-1} [34].

The Laplace operator eigendecomposition gives us sufficient structure to define filters.

$$g(\Delta) = V^T g(\Lambda) V \tag{2.35}$$

Relating this back to PDE's and their spectral decomposition, we see that in this case $g(\lambda) = e^{-\lambda}$, corresponding to the heat kernel, is a low-pass filter because all high-frequency components are exponentially decayed.

3 METHODS

We first introduce our dataset and the signal we wish to estimate: the wall shear stress (WSS) and blood pressure (BP). We also describe and motivate our choice of input features for the networks.

Then, we give an overview of the different neural network structures, and motivate our architectural choices. In order to predict the WSS signal on meshes, we chose to use DiffusionNet [35] as a baseline, since all networks we create are derived from it. In particular, we introduce derived networks using parallel wavelet based feature evolutions, and a specific class of wavelets that evolves the feature state via the dynamics of the wave equation.

3.1 Dataset

3.1.1 Arterial Mesh Dataset

We use a synthetic dataset of coronary artery meshes created by [38] [39]. The meshes generated reflect properties of human coronary arteries. Both a dataset with single and a dataset with bifurcating arteries is available.

From the synthetic mesh, the WSS and BP are computed by [38] using computational fluid dynamics (CFD). The BP is expected to be a global signal, dependent on the shape of the artery, whereas the WSS is mostly related to local vessel geometry [39]. The bifurcating artery is expected to have a more challenging WSS to compute, as the dynamics of the fluid inside are more complicated. We therefore choose to estimate biomarkers only on the bifurcating arteries.

The bifurcating coronary artery dataset contains 2,000 samples. We split the dataset randomly into train, validation sets, keeping the test set fixed, via an 8:1:1 ratio. The split occurs randomly per experiment, therefore each trained network has different train and validation sets from other networks. However, the test set is kept the same for all experiments.

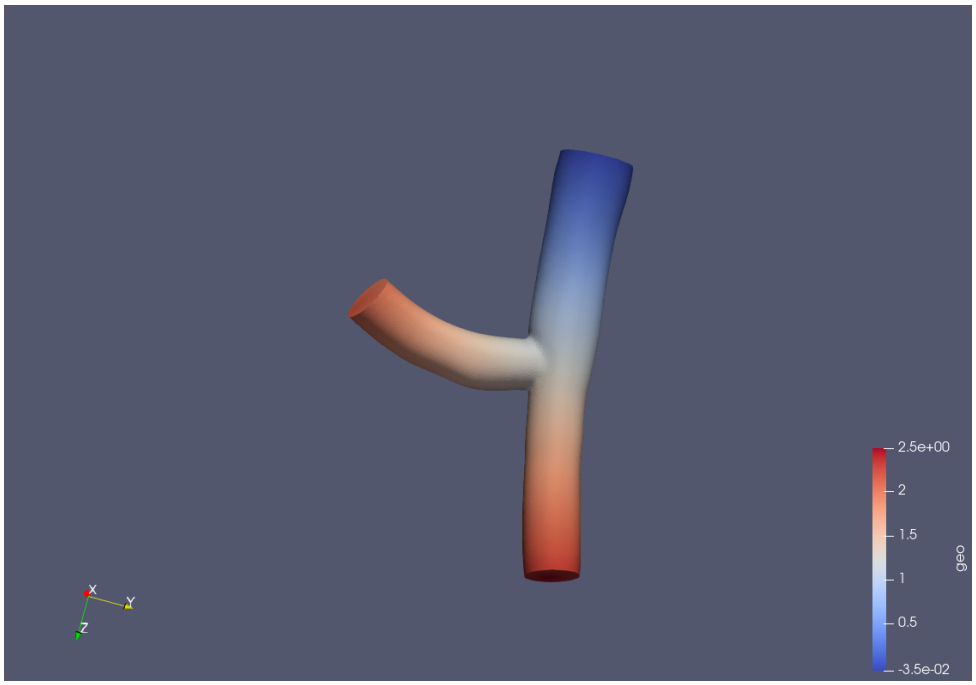


Figure 3.1: Visualization of geodesic distance to inlet. Areas coloured blue are close to the inlet, areas coloured red are further.

3.1.2 Feature Description

We compute features intrinsic to the artery surface. A notable exception is the geodesic distance to the inlet (Figure 3.1), where we permit ourselves the knowledge of where blood enters the artery. Additionally, we use several features invariant to rigid transformation.

We say a signal f is invariant to rigid transformation, a composition of rotation and translation:

$$T_{\theta,x} f = \begin{pmatrix} R_\theta & T_x \\ 0 & 1 \end{pmatrix} f = f \tag{3.1}$$

when the signal value at any node remains unchanged if the surface mesh is rotated or translated. Since PDEs such as the heat and wave equation are invariant to rigid transformations applied to the surface, the kernel signatures also are. This can be most easily verified by the fact that the Laplace-Beltrami operator itself is an invariant operator, since it is intrinsic to the shape.

Invariance to rigid transformations can be a benefit: properties intrinsic to the shape of the vessel such as the blood pressure do not depend on the shape's orientation. Therefore, using features that depend on the shape's orientation to predict the BP may cause the network to overfit. However, the WSS being a vector quantity, it is defined relative to some coordinate frame. This means a random rotation or translation will change the WSS signal because the frame of reference is now changed. Thus the WSS is equivariant to rigid translation: while the actual vectors stay the same, because we observe them through an origin, the representation does change under rigid transformation. Using invariant features removes the network's ability to distinguish between rotated features.

- **Geodesics:** to help the network understand the direction of blood flow in the artery, we define the *inlet* to be where blood flows into the artery in accordance with the input features used in [39]. For each vertex this geodesic distance is computed to the nearest inlet

vertex using [1]. The geodesic distance between vertices, or more generally points on the arterial surface, is defined by the length of the path over the surface that connects them, making it different from the Euclidean distance. The geodesic distance is invariant to rigid transformation.

- **Mesh Normals:** this feature provides information about where the exterior of the mesh is. At each vertex, a vector can be defined that is normal to the smooth surface. The normal generated is based on the faces neighbouring the node. The mesh normals, being a vector quantity, are equivariant to rigid rotation.

Additionally, we use two features related to PDE's on meshes: the heat kernel signature [40], and the wave kernel signature [1]. We use these features to provide information on the shape's intrinsic properties by characterizing the vertices by the evolution of heat and wave based processes on the surface. They are invariant to rigid transformation, and offer information at multiple scales.

3.1.3 Kernel Signatures

Kernel signatures are feature descriptors based on the kernels of PDEs that describe processes on the surface, These processes are dependent on local and global properties of the surface, making these features useful in encoding information about the geometry of the surface or shape.

The heat kernel signature (HKS) [40] and wave kernel signature (WKS) [1] are kernel-based feature descriptors invariant to orientation preserving isometric deformations, and when used as network inputs can make the network less sensitive, though not invariant, to rotation. [35] We use these feature descriptors to provide compressed information about the shape beyond its local geometry into the vertex features.

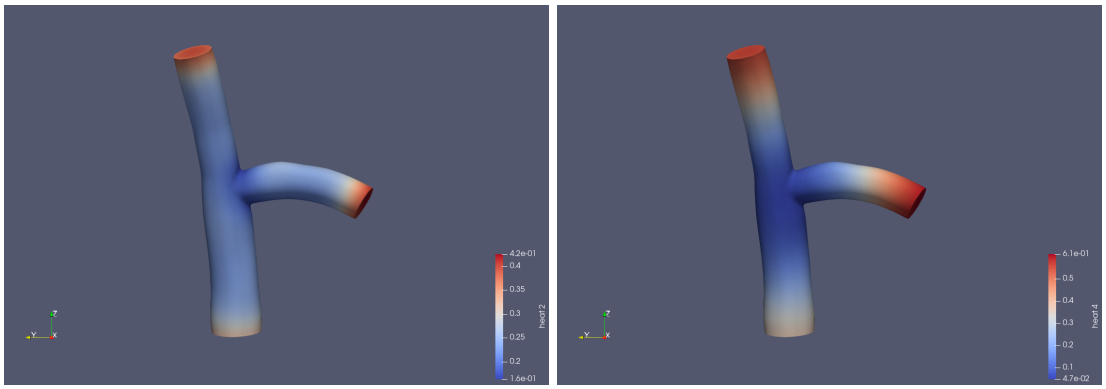
Heat Kernel Signature (HKS): given by the heat kernel distance between two points, it represents the amount of diffusion that occurs from a given point after some time t.

$$\text{HKS}(x, y, t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i(x) \phi_i(y). \tag{3.2}$$

The HKS is multi-scale: for small t the value of the signature at a point x is primarily influenced by the local geometry, the reverse is true when t is large. [40] This signature provides the network with features created from both local and global geometric properties of the surface.

The HKS can be interpreted as the probability density function for Brownian motion on the manifold. It is invariant under isometries (and thus rigid transformations), and stable under small deformations of the manifold. This stability to deformation can be motivated by interpreting the $\text{HKS}(x, y)$ as the average of all paths possible between x and y in time t. [40]

[1] Potpourri3d



(a) HKS scale 2 (b) HKS scale 4

Figure 3.2: HKS visualized at scale 2 and 4 of a logarithmic timescale. As the scale and thus t increases, more heat leaves each vertex. Vertices in regions with low mesh connectivity retain more heat, visualized in red.

Wave Kernel Signature (WKS): [1] following a similar principle to the HKS we may define the WKS for a vertex x

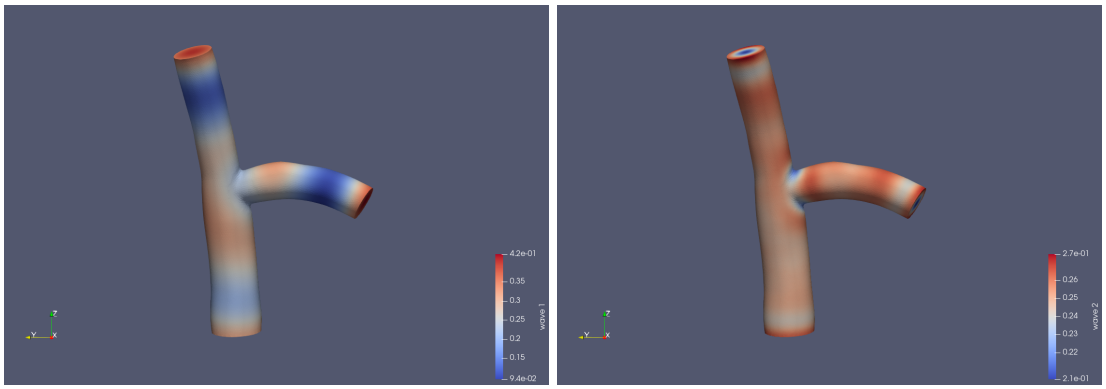
$$\text{WKS}(x, e_0) = C_e \sum_{k=0}^{\infty} \phi_k(x)^2 e^{\frac{(-e_0 - log(E_k))^2}{2\sigma^2}}. \tag{3.3}$$

The WKS is inspired by the quantum mechanical Schrödinger wave equation

$$\frac{\partial u}{\partial t} = i\Delta u \tag{3.4}$$

and can be interpreted as the probability of a particle being at a particular location on the mesh, given an unknown initial position of the particle, and some initial measurement of its energy e_0. The WKS differs from the HKS by not being time-dependent, instead the scale is the energy level.

For the WKS a physical interpretation of scale also exists: particles with high energy are able to travel out of areas more quickly, lower energy scales display confined spaces where particles might become trapped. [1] This can be visualized in Figure 3.3



(a) WKS scale 1 (b) WKS scale 2

Figure 3.3: WKS visualized at scale 1 and 2 of a logarithmic timescale. The scales represent energy levels of a quantum-mechanical particle with some unknown position. The intensity values represent the probability of finding a particle in a given area. Particles with high energy are able to travel out of areas more quickly, lower energy scales draw confined space where particles might become trapped. [1]

3.2 Structure of the data

In this section we explore the biomedical data we are using, in order to better understand the challenges the network faces.

Blood Pressure: we know pressure is a force applied to an area: $P = \frac{F}{A}$. Although pressure is represented as a scalar quantity, this scalar is understood to be the magnitude of the force normal to the area the pressure acts on. Pressure is a force acting on some area, the direction of the normal vector of that area. The blood pressure (BP) is around the order of magnitude of $1.34 * 10^5$ millimeters of Mercury (mmHg). We rescale the pressure to Hg to prevent numerical instabilities in the methods. The blood pressure at any point is given by a scalar, which is understood implicitly to be the magnitude of the normal vector at that point on the surface.

Wall Shear Stress: the WSS is a vector quantity of varying magnitude and varying direction that is a result of movement of fluid parallel to the artery walls. We know that the WSS is related to local geometry. We see in the image below an example of the x-coordinate of the WSS. Note that the signal is decidedly smooth on most of the surface, however there are key parts where the signal changes rapidly on points geodesically close to each other.

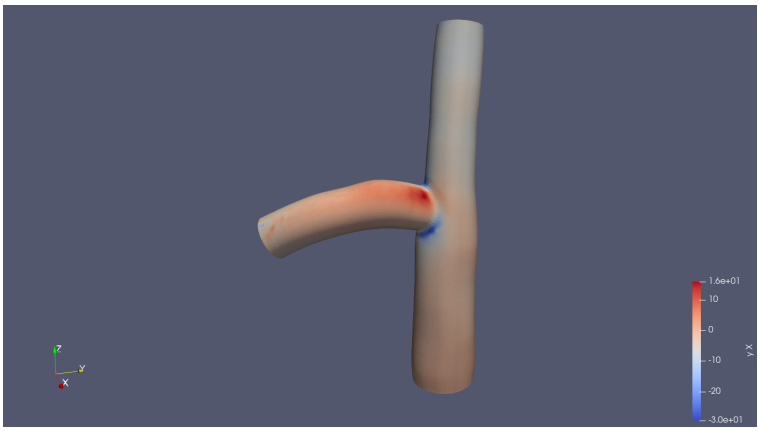


Figure 3.4: Magnitude of the WSS signal. The WSS is a vector quantity at each point on the surface. WSS vectors in areas near the bifurcation are subject to rapidly changing magnitude

In order to better understand the spectral makeup of the WSS, we project the signal to a spectral basis.

Spectral components & discretization: we can construct an orthonormal basis of eigenvectors of the cotangent Laplacian [29], a common discretization of the Laplace-Beltrami operator on manifolds represented by meshes. The cotangent Laplacian is given by

$$\mathcal{L}_{i,j} = \begin{cases} \frac{1}{2}(\cot(\alpha_{ij}) + \cot(\beta_{ij})), & \text{if } (i,j) \in \mathcal{E} \text{ and } i \neq j \\ \sum_{j \in N(i)} w_{ij}, & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

where α and β are the angles that are opposite the edge (i, j). Any edge can only ever be part of two triangle faces, and the angles not connected to the edge (i, j) are α and β respectively.

The matrix $\mathcal{L}$ is symmetric and positive semi-definite, allowing us to easily find its eigenbasis V and diagonal matrix of eigenvectors Λ as

$$\mathcal{L} = V^T \Lambda V \tag{3.6}$$

The implementation uses DiffusionNet's eigenvector estimation procedure, which in turn uses the σ shift method implemented by Scipy [44]. Via this method, which requires a positive semi-definite symmetric matrix, we can search for the eigenvectors in order of magnitude of their eigenvalue. This way, we can compute the first k eigenvectors without needing to compute the whole spectrum. The resulting eigenvectors are orthogonal with respect to M[35], i.e. $\langle \phi_i, \phi_j \rangle = \phi_j^T M \phi_i = 0$ when $i \neq j$, and normalized via $V^T M V = 1$ [35].

We see the eigenvalues of the mesh are increasing in a linear way. The eigenvalues are all real and positive, and the eigenvectors corresponding to low eigenvalues are spatially smooth, whereas the higher eigenvectors of the cotangent Laplacian are spatially variant.

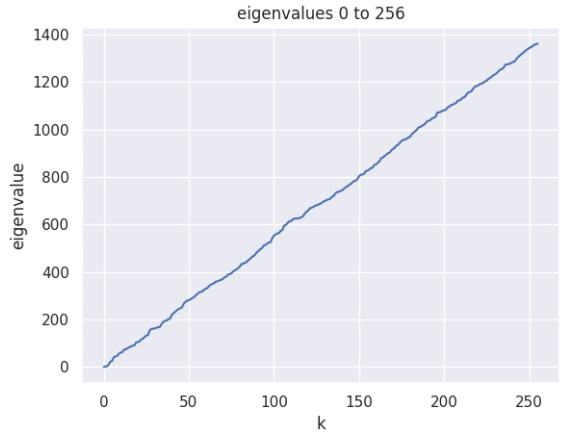
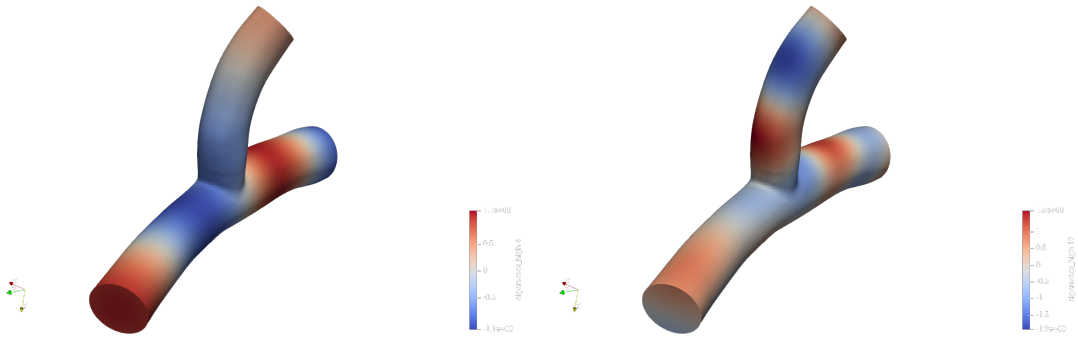


Figure 3.5: Plot of k vs λ_k. The eigenvalues are increasing monotonously and never negative.

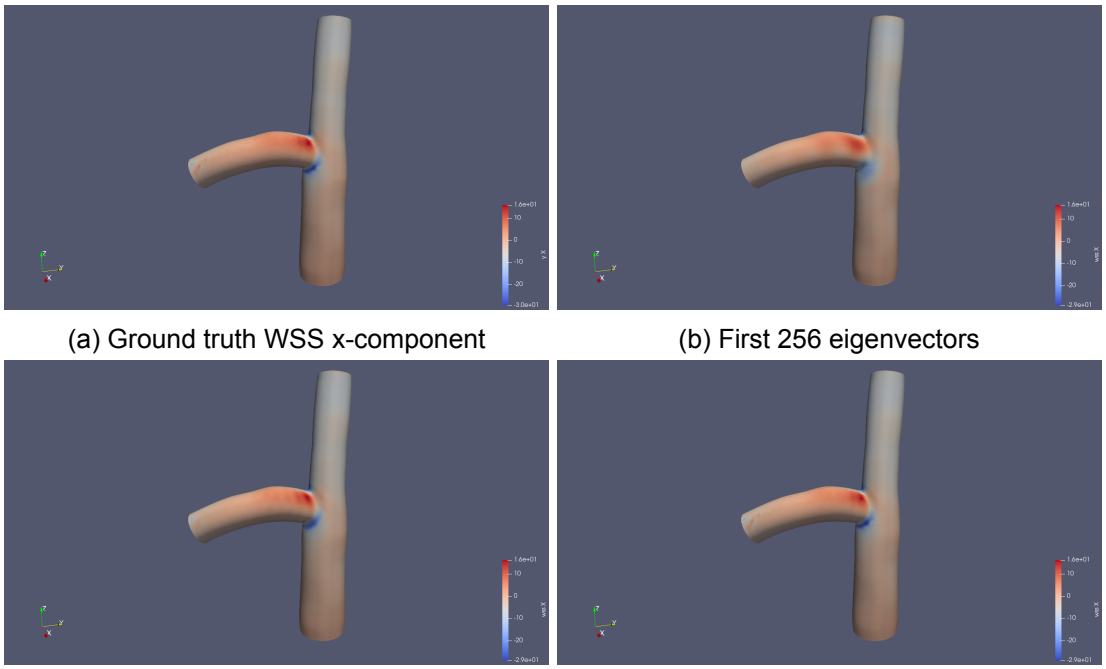


(a) 5th eigenvector (b) 20th eigenvector

Figure 3.6: The x-component of the WSS is projected onto spectral bases of different number of eigenvectors. The signal, expressed in the reduced bases, is then projected back to the vertex domain. We see the jittery nature of the WSS signal is most present in the reconstruction using the higher bases.

Spectral decomposition of WSS: we aim to discover which of the eigenvectors are best at explaining the WSS signal. Since the eigenvectors are we orthogonal, the change of basis V already expresses the best projection of each eigenvector onto our WSS signal. Additionally, since the eigenvectors are normalized via $\phi^T M \phi = 1$ [35] we may also assume that we can interpret the spectral coefficients via equal weighting (i.e., one vector does not have significantly higher norm).

In the following plots we show the WSS signal projected onto spectral bases of varying number of eigenvectors, to investigate to what extent the high frequency eigenvectors are responsible for the WSS signal. This is especially informative when considering DiffusionNet is, spectrally, a low-pass filter. If the WSS contains a large amount of high-frequent information, diffusion alone may not provide sufficient information sharing.



(a) Ground truth WSS x-component

(b) First 256 eigenvectors

(c) First 1700 eigenvectors

(d) First 17,000 eigenvectors

Figure 3.7: The x-component of the WSS is projected onto spectral bases of different number of eigenvectors. The signal, expressed in the reduced bases, is then projected back to the vertex domain. We see the jittery nature of the WSS signal is most present in the reconstruction using the higher bases.

Projecting the signal onto its lower-dimensional eigenbasis amounts to a compression problem, or alternatively, a low-pass filter. In Figure 5.2 we see that 256 eigenvectors results in a very smoothed out version of the signal, missing crucial details such as the dip near the end of the left-most part of the vessel. This is better represented in the reconstruction using 1700 eigenvectors, suggesting at least some of the higher frequency, more spatially variant eigenvectors play a role in the WSS signal. This is akin to the role of high frequent components in the spectral domain of images, which often represent sharp edges or other details. In the WSS, the high frequent components play a similar role, and while most information resides in the lower end of the frequency spectrum, the high frequent features are still important for detail.

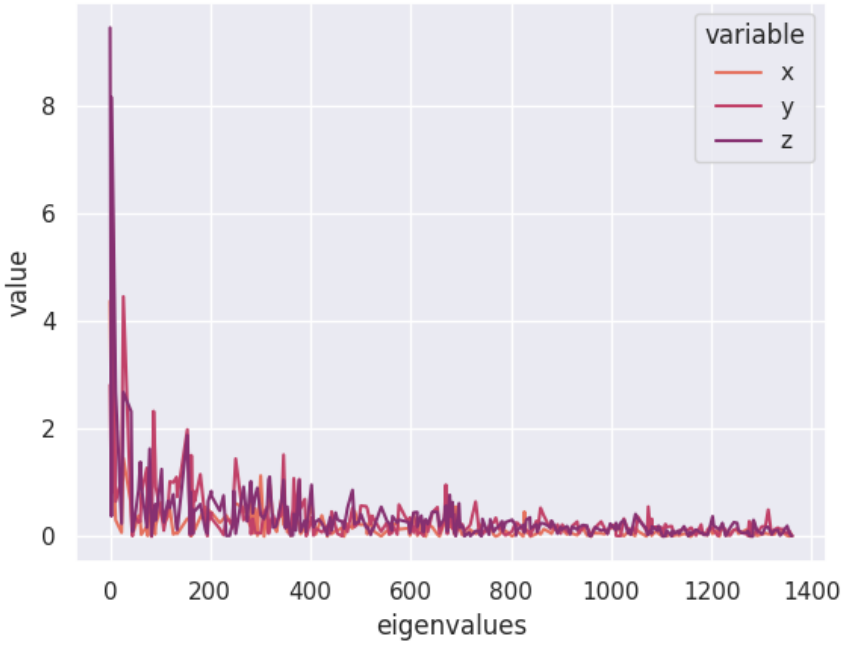


Figure 3.8: Projection of WSS signal onto spectral basis of 256 eigenvectors. Plotted are the absolute values of the basis coefficients needed to represent the WSS signal. On the x-axis is the eigenvalue of the eigenvector associated with that coefficient, i.e. the last entry, eigenvector 256, has λ_{256} around 1400. We see that the majority of the information is captured by the lower end of the spectrum, as visually verified by Figure 5.2

.

3.3 Wave Dynamics

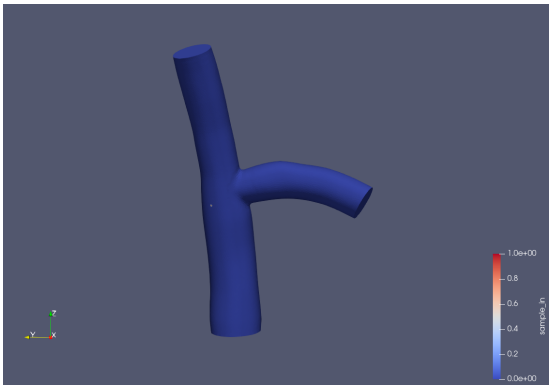
Where DiffusionNet uses the heat equation to update the feature vector and share information, we are interested in using the wave equation. This is due to the fact that diffusion favours information sharing between close nodes, and from a spectral perspective removes detail. We show that the wave equation does the opposite, allowing distant nodes to share their features, propagating the vector via wave dynamics, and from a spectral perspective leaving certain high frequency eigenvector bands active in the signal.

By wave dynamics we mean a movement of features from areas of high intensity to areas of low intensity, typically transporting the majority of the mass or energy of the wave away from the initial location.

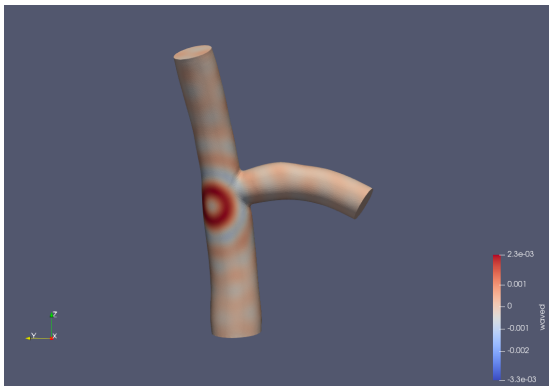
From a spatial perspective we investigate the long-range dynamics visually via the evolution of a point source (Figure 3.9a), and the evolution of one of the input feature vectors (Figure 3.10). The long-range dynamics of the wave equation can be visualized via this point source in Figure 3.9a. We see in the image the point source, i.e. the mesh signal being zero at each vertex except one, where it has a value of 1. The corresponding contribution of the feature of this node on the feature of other nodes is visualized in the image.

The point source in Figure 3.9a emits a wave propagating outward over the surface, with signal intensity appearing highest around the wavefront. We also see several other wavefronts, which

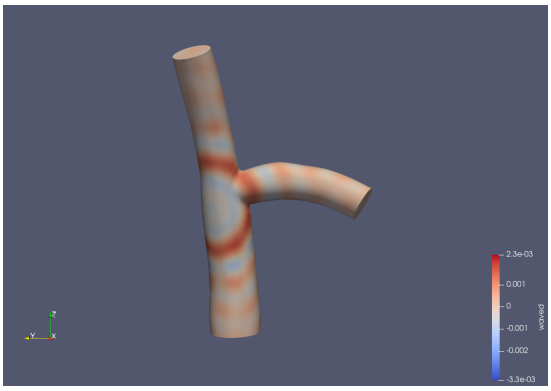
are likely artifacts of the reduced eigenbasis and not intended behaviour. Additionally, the legend is not symmetrical, which could point to a more negative weighting of the blue ring around the initial red ring than apparent from the image. Still, the kernel selects a receptive field that appears primarily influenced by the wavefront.



(a) Location of point source (red dot)



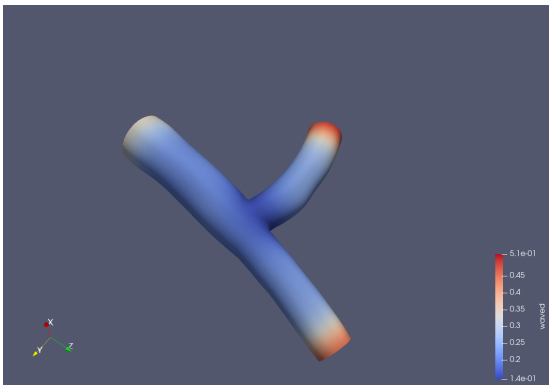
(b) Point source at t_1 under wave dynamics



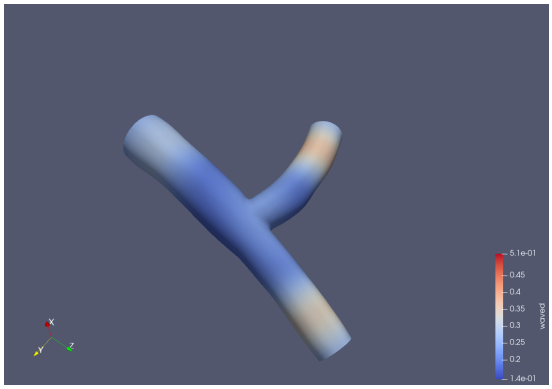
(c) Point source at $t_2 = 2t_1$ under wave dynamics

Figure 3.9: A point source is propagated using a reduced basis of 256 eigenvectors via the wave kernel. In stark contrast to diffusion, via the wave equation the contribution of the point source's feature can skip over neighbours, defining a ring where the wave interacts the most, visualized in red.

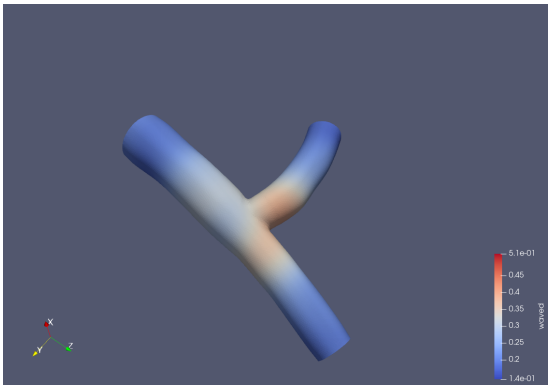
In Figure 3.10 the wave equation is used to propagate an initial feature u forward in time. We can interpret the feature evolution as follows: suppose there is a liquid covering the vessel. Areas where the initial feature u is red correspond to the liquid being lifted in its initial position, and the blue areas where it is pushed below its resting position. Then the initial feature u is evolved until time t, and the perturbation travels in a wavelike manner to the center of the vessel, where it meets the other waves caused by the initial perturbed state u, and gets reflected back. We can interpret this movement of the feature vector as constructive interference of multiple point sources: nodes with positive displacement that are close to each other will create wave travelling radially outward.



(a) Initial feature distribution



(b) Feature after propagating the wave equation for 10 timesteps



(c) Feature after propagating the wave equation for 26 timesteps

Figure 3.10: An initial feature (a) is propagated for different amounts of time. The amplitude of the feature is conserved from (b) to (c). The wave equation provides a way to propagate features over the mesh, to distant nodes. This shows the wave equation should be a good candidate for long-range message passing over a manifold. The wave equation is evaluated using the spectral wave kernel [18].

3.4 Spectral Wave Equation

Diffusion reduces detail in a signal and smooths it out, because it is a low-pass filter via its kernel $e^{-\lambda t}$ exponentially decreasing the contribution of high-frequency eigenvectors. On the other hand, the wave kernel is a band-pass filter, allowing it to preserve details of the signal.

The spectral wave kernel we use is derived by [18]. The kernel is based on the wave equation with zero initial velocity

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = -c^2 \Delta u, \quad c \in \mathbb{R} \\ \frac{\partial u}{\partial t}|_{t=0} = 0, \\ u|_{t=0} = x \end{cases} \tag{3.7}$$

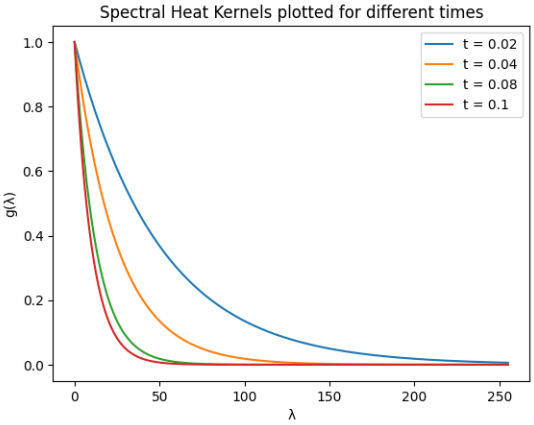
In other words, given some initial perturbation x, and confining our PDE to an initial velocity of zero, we find the solution of the dynamics after t seconds. This solution represents a movement rather than diffusion of features: the features move across the surface in a wave-like manner. Since the mesh is a surface without a boundary, we do not have any initial boundary conditions to take into account.

Because we are solving the wave equation on the mesh, we use the eigenfunctions of the cotangent Laplacian as our spectral basis, and try to express the solution operator in this basis.

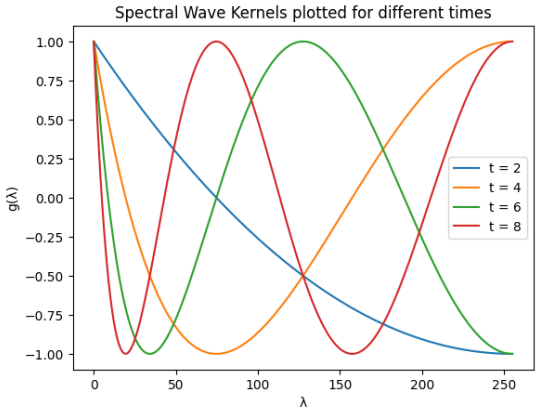
$$\begin{cases} u(t) & = V g(\Lambda, t) V^T \\ g(\lambda_l, t) & = \cos(t \cos^{-1}(1 - \frac{s\lambda_l}{2})) \end{cases} \tag{3.8}$$

In the equation above, $g(\lambda_l, t)$ represents an element-wise scaling of the entries of the diagonal matrix of eigenvectors Λ. The matrix V is the stacked matrix of eigenvectors, s is the wave speed, and λ_l the eigenvalue of the corresponding eigenvector v_l in V.

This wave kernel $g(\lambda_l, t)$ is stable only for wave speed $s < \frac{2}{\lambda_m}$ where λ_m is the largest eigenvalue of the eigenbasis. Additionally, the initial wave speed is taken to be zero. Although theoretically the wave operator can be expressed without assuming an initial velocity of zero, [3], the resulting operator is numerically unstable.



(a) Spectral heat kernel $g(\lambda) = e^{-\lambda t}$ plotted for different λ



(b) Spectral wave kernel $g(\lambda_l, t) = \cos(t \cos^{-1}(1 - \frac{s\lambda_l}{2}))$ plotted for different λ

Figure 3.11: Comparison of spectral heat and wave kernels, for the same λ. Depending on the value of t, the heat kernel drops coefficients for high frequency vectors. The wave kernel does not do this, only dropping coefficients in certain frequency bands.

As seen in Figure 3.11, the kernel when plotted against time shows that different parts of the frequency spectrum get activated at different times, much like a band-pass filter. Compare this to a similar image of the diffusion kernel, which is a strictly decreasing exponential, and thus the importance of high frequency components of features is always strictly lower than the low frequency components of features.

As such, wave kernel can establish long-range interaction between vertcies, and spectrally preserve detail within the signal.

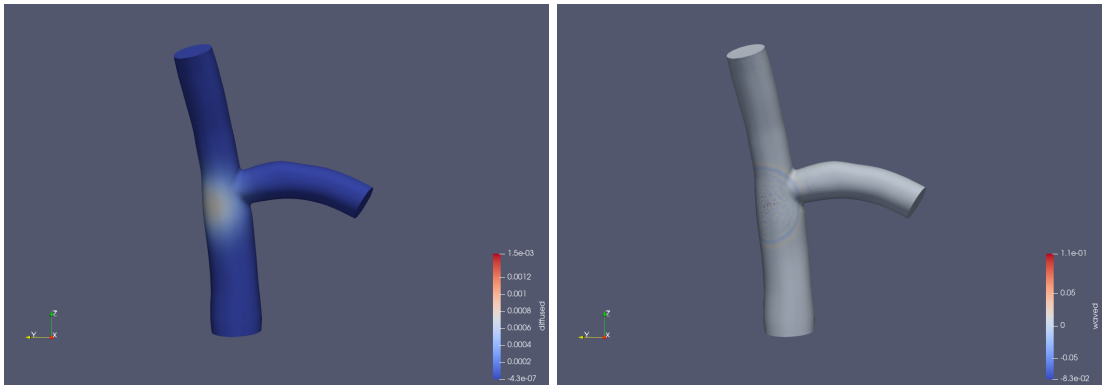
Additionally, a spectral solution is computationally inexpensive, a very important property for deep learning methods. We see this partially in DiffusionNet's own paper, where large meshes (20,000+ vertices) are not feasible to solve via the implicit method on the GPU due to memory issues and computation time constraints. Because we wish to solve the wave equation for each feature channel, and a typical DiffusionNet implementation uses around 100-200 channels, we are evolving many initial feature vectors to different times t.

The spectral solution is so computationally inexpensive because there is no system that needs to be solved: unlike implicit methods, we do not need to solve a large system of equations, via e.g. a cholesky decomposition. In our experiments, the implicit method for the heat equation crashed because the resulting matrix problem took too much memory (over 20gb) on the GPU. Conversely, networks using the spectral method use

$$u(t) = V^T g(\Lambda) V \tag{3.9}$$

which amounts to matrix multiplication, once the eigenbasis V of the mesh is found. The eigenbasis is unique to the mesh, so it needs to be computed only once per mesh, accelerating training. We may use a reduced basis of eigenvectors. This sacrifices accuracy of the PDE evaluation, but reduces both computation time of the PDE solution and pre-computation time of the eigenbasis.

Reduced basis: contemporary works using spectral diffusion compute only the first k vectors of the eigenbasis. For diffusion this is natural since it is a low-pass filter, however for the wave equation this results in some artefacts, as seen in 3.12



(a) Point source at $t_2 = 2t_1$ under wave dynamics using full eigenbasis

(b) Point source at $t_2 = 2t_1$ under wave dynamics using full eigenbasis

Figure 3.12: The above evolution of the point source was calculated with an eigenbasis of 17,000 eigenvectors. We see that, in comparison to the reduced basis, the wave contains a lot less oscillations, and the displacement is strictly localized around the center of the point source. For diffusion, the image is indistinguishable from those calculated with the reduced eigenbasis.

However, as seen in Figure 4.6 where features propagate over the mesh, in aggregate we still observe wave dynamics even with the reduced eigenbasis. These properties could make WaveNet more equipped to reconstruct high frequency components of biomarkers than DiffusionNet.

3.5 Network Architectures

In this section we introduce the three main architectures we use: WaveNet, WaveDiffusionNet, and Parallel PDE Channels (PPC). First, we show a building block shared by all of these networks: the wave block.

A wave block follows the same structure as a diffusion net block. It computes the solution of the wave equation propagating until time t, for the initial feature map $\mathcal{F}$. The solution of this PDE

is found through the spectral form of the wave equation. This is implemented through using the spectral wave kernel, rather than the spectral heat kernel.

WaveNet follows the same structure as Diffusion Net. The primary difference is the spectral kernel used to evolve the features at each layer: in the spatial diffusion layer, instead of multiplying each eigenvector's contribution by $g(\lambda_l, t) = e^{-\lambda_l t}$, we multiply by $g(\lambda_l, t) = \cos(t \cos^{-1}(1 - \frac{s\lambda_l}{2}))$.

Like DiffusionNet, in WaveNet the spatial gradient of the evolved feature u_t is calculated, since the wave kernel suffers from the same isotropy as the diffusion kernel. Structurally, the network is identical to DiffusionNet because many of the operating principles have remained the same: we use an isotropic PDE with a spectral representation to evolve the feature vector to some time t.

WaveDiffusionNet Although the wave kernel offers many benefits in terms of long-range communication, in some cases sharing information with neighbours is also important. For this reason we consider a network where the feature vector u_0 is first diffused via the wave equation and then propagated via the heat equation, i.e. letting t_w and t_h be learned parameters for the heat and wave equations respectively, then $u(t_w, t_h) = \cos(t_w \cos^{-1}(1 - \frac{s\lambda_l}{2}))e^{-\lambda_l t_h} u_0$. Our motivation for combining the kernels is as follows: if DiffusionNet wants to share information between distant neighbours, the feature vector gets diffused to a large amount. With a combination of wave and diffusion dynamics, the network is free to use the propagational element to define long-range interactions, while using diffusion to spread the signal over the distant region.

Parallel PDE Channels Net (PPCNet) Where DiffusionNet diffuses a feature channel to a time t which is learnt specifically for that channel, Parallel PDE Channels Net (PPCNet) applies different PDEs with different learnt evolution times to the channel. By evolving the same initial feature to different learnt times and even via different PDEs, PPCNet is able to define long (wave) and short (diffusion) interactions, and additionally optimize the time for each of those interactions. This principle can be seen in Figure

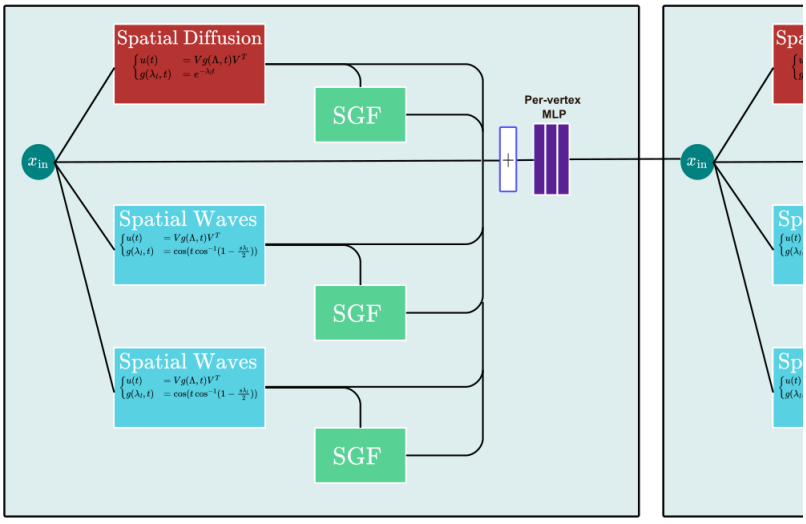


Figure 3.13: Illustration of PPC network architecture. The network applies different PDEs to the same channel, providing the MLP with multiple forms of spatial information. Not shown is the skip connection from the input features to the output of the MLP.

We then find the spatial gradient features for each of these PDE evolutions of the channel. All of the spatial gradient features and the evolved channels are concatenated into a single tensor,

which the per-vertex MLP takes as input. We do this so the MLP has a richer form of spatial information than one diffused signal, instead having access to multiple evolutions of the same feature channel.

Another way we might motivate PPC is by comparing DiffusionNet to a transformer network. Specifically, the structure of DiffusionNet resembles the encoder of a transformer: both a transformer and DiffusionNet use multiple encoding layers stacked in series, each with skip-connections, linear layers, and attention. Within DiffusionNet the attention comes from the diffusion PDE, which is best interpreted via the Figures of point-sources we show in the previous section, Figure 3.9a and 2.7. In these sections, a particular weighting is attached to the features of every vertex on the mesh (including the point source's own feature). This can be viewed as a form of spatially regularized attention, where the importance we assign each mesh vertex's feature is based on the diffusion or wave equation.

Where DiffusionNet uses single-head attention via allowing each channel a single attention map, in PPC we implement a form of multi-head attention, giving each channel the ability to specify multiple attention maps via the different PDEs and evolution times.

3.6 Network Hyperparameters

We train our networks using PyTorch [28] and Pytorch Geometric [15] For each network, we list the most important hyperparameters. We also list some general hyperparameters all networks use.

All networks have a hidden layer of size 200. In PPCNet this hidden layer size results in the first layer of the per-vertex MLP containing more input nodes, because we concatenate the output of multiple Spatial Gradient Feature modules and Spatial Diffusion modules and feed this to the input. In this case, PPCNet has $7N$ input channels for the MLP, where N is the hidden size. DiffusionNet on the other hand has $3N$: the skip connection, spatially diffused signal, and spatial gradient features.

We split the learning rate for general weights and biases in the network and the learned times. The former use a learning rate of $3e-4$, and the latter of $3e-3$. This is done to encourage the network to investigate several different realizations of the PDE process.

Additionally, all networks use the HKS and WKS, each initialized using a logarithmic scale and taking the first 4 scales. The code implementation was found in [cite dominik pesh(?)]. We use 256 eigenvectors to compute the HKS, WKS, and PDE solutions. Finally, we train for 200 epochs.

Model specific parameter choices: we initialize DiffusionNet at 0.0, as the authors do. For WaveNet, we choose to initialize one network, WaveNet-low, to $t = 1.5$ and one to $t = 3.0$. This choice was based on investigating point source evolution for several times and selecting a promising radius. The higher time was intended to be biased towards long-range tasks such as the blood pressure.

In our choice of initial t we avoid choosing t too small, since this somewhat resembles diffusion, and we want to ensure the performance comes from using the wave kernel.

Networks	Initial Time
WaveDiffusion	Wave: 3, Diffusion: 0
DiffusionNet	Diffusion: 0
WaveNet-low	Wave: 1.5
WaveNet-high	Wave: 3
PPC	Diffusion: 0, Wave: 3, Wave: 5

Table 3.1: Initial time parameters for all networks used.

3.6.1 Oversmoothing

Conventional deep learning techniques often use deep network structures, of multiple layers, to facilitate a large number of parameters in their network. A network with more parameters is generally able to represent more complex functions, via the Universal Approximation Theorem.

However, conventional message-passing GNNs suffer from oversmoothing [31] [27]. This phenomenon causes all vertex features of a layer to become uniform after a certain amount of layers, preventing the construction of deep GNNs.

Since DiffusionNet contains a spatial diffusion module that acts as a low-pass filter, smoothing out the features, oversmoothing via stacking too many layers might be a problem for DiffusionNet too. We can measure how diverse the signal is through the Dirichlet energy of the vertex features [31].

Definition 3.6.1 (Graph Dirichlet Energy)**.** Let $\mathcal{N}$ denote the set of nodes for graph $G = (\mathcal{V}, \mathcal{E})$, and let X_i be the value of a node feature at node $i \in \mathcal{N}$. Then the Dirichlet Energy DE is given by

$$\text{DE(G)} = \frac{1}{v} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} ||X_i - X_j||^2. \tag{3.10}$$

If the Dirichlet energy of the node features becomes low, it shows a lack of variation in the information, pointing to oversmoothing [31][32]. We can thus measure the Dirichlet Energy of the input features after each layer of a trained DiffusionNet. If the Dirichlet Energy goes down each layer, this could point to issues of oversmoothing within DiffusionNet

3.7 Representational Power of Networks

In order to better understand the capacity of WaveNet and DiffusionNet to represent detail within the signal, we train the network to reconstruct the WSS of one particular mesh. The WSS of this mesh was high-pass filtered (the first 178 eigenvectors were dropped). We set to zero the coefficients related to the first $178 = 0.7 \cdot 256$ because an eigenbasis of 256 is able to explain most of the signal, as seen in Figure 4.6, suggesting the coefficients of the first 256 vectors contain at least a large share of the low frequency information. We therefore filter out this low frequency information by:

$$u_{\text{highpass}} = (u - (V^T G(i) V M u) \tag{3.11}$$

This way, we can see to what extent each network can fit a function to a high frequency signal.

By using the spectral wave kernel, which lets through more high-frequency information than the diffusion equation, we expect WaveNet to outperform DiffusionNet in the reconstruction of detail

within the signal. In order to test this, we examine each network's ability to overfit to one train sample.

Specifically, we remove all low-frequency eigenvectors from the signal, to keep only the detail of the signal. We then train and test on the same one mesh. The networks we test are: DiffusionNet, WaveNet (with high and low initial time), and PPC.

If the networks are not able to fit perfectly to the train sample, this would mean their representational capacity is low and they are not suitable for learning the signal.

Similarly, we train the above networks on a small dataset of 50 meshes containing only the high frequent parts of the WSS signal (the first 178 eigenvectors were dropped). We drop $178 = 0.7 \cdot 256$ because our eigenbasis of 256 is able to explain most of the signal. The low frequency information seems to reside in this range. We therefore filter it out by:

$$u_{\text{highpass}} = (u - (V^T G(i) V M u)) \tag{3.12}$$

where our lowpass filter is given by the diagonal matrix $G(i,i) = 0$ except for $i < 178$, where $G(i,i) = 1$. In other words, from the original WSS we subtract a filtered WSS containing only its lowest frequencies. By subtracting this low-pass WSS, we remove all low frequency content, creating $\text{WSS}_{\text{highpass}}$.

We evaluate this experiment only via the estimated magnitude of the WSS signal, since the components are correlated to the magnitude.

3.8 Choice of Loss Function

Because the WSS is a signal that varies heavily spatially, we wish to ensure that not capturing such spatial outliers is heavily punished. The L_1 loss function, used as our default loss for all networks, is by design poor at punishing outliers, favoring networks that stay close to the mean of the signal. We therefore investigate DiffusionNet with L_2 loss, to see if changing the loss function could lead to more detail present in the highly spatially variant areas of the reconstruction.

3.9 Evaluation metrics

The WSS vector is estimated component wise: the network outputs a 3D vector at each vertex. In order to compare these vectors to the ground truth WSS vectors we use the mean average error (MAE) and mean cosine similarity (MCS), similar to [39]. The MAE is a measure of how different the label and estimated WSS vectors are. The MCS is a supplementary metric that measures whether the label vector y and estimated vector x point in the same direction. Although the MAE is able to estimate the global estimation error, the MCS allows us to distinguish between errors in vector orientation and vector magnitude. If the MCS is low and the MAE is high, this would mean we estimate the correct direction of the WSS vector, but under- or overshoot in our prediction of the WSS vector magnitude.

Definition 3.9.1 (Mean Average Error)**.** Let , then the mean average error (MAE) is given by

$$\text{MAE} = \frac{1}{N} \sum (x - y)^2. \tag{3.13}$$

Definition 3.9.2 (Mean Cosine Similarity)**.** Let , then the mean cosine similarity (MCS) is given by

$$\text{MCS} = \frac{1}{N} \sum \frac{(x \cdot y)}{|x||y|}. \tag{3.14}$$

4 RESULTS

4.1 WSS estimation: quantitative results

DiffusionNet with L2 loss performs best, followed closely by DiffusionNet with L1 loss, and finally WaveDiffusion. Both WaveNet-low and WaveNet-high diverge during training on the WSS.

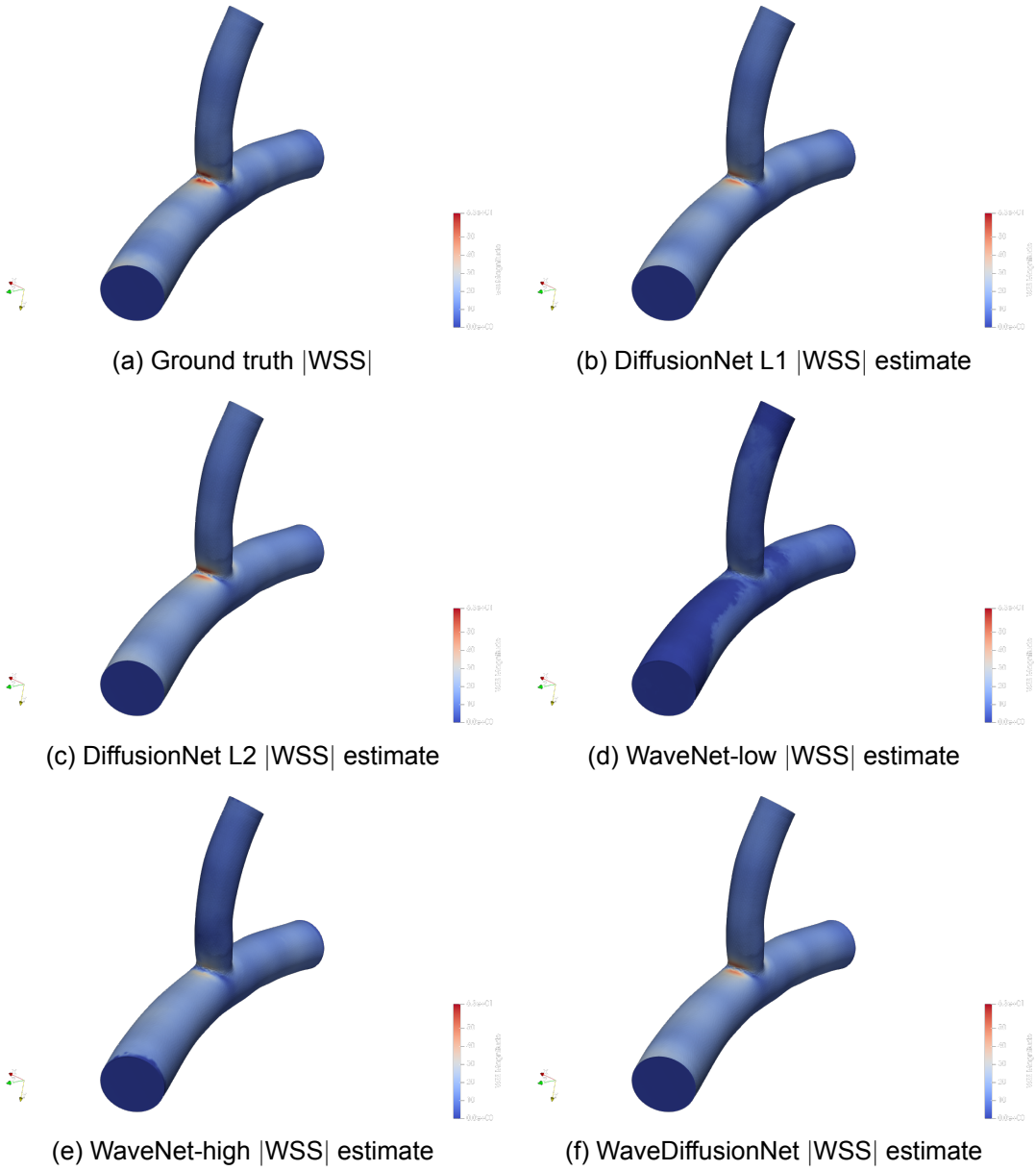
WSS Networks	NMAE	Approximation Error	Mean Cosine Similarity
WaveDiffusion	0.6 - 0.3	13.1 - 3.9	1.00
DiffusionNet L1	0.6 - 0.3	12.3 - 3.7	0.90 - 0.01
DiffusionNet L2	0.6 - 0.2	12.1 - 3.1	0.90 - 0.01
WaveNet-low	3.6 - 0.9	63.7 - 6.7	0.79 - 0.05
WaveNet-high	1.5 - 0.5	30.2 - 7.9	0.88 - 0.03
PPC	3.0 ± 0.6	47.6 ± 3.8	0.87 ± 0.2

Table 4.1: WSS reconstruction metrics for all networks. DiffusionNet with L2 loss performs best, followed closely by DiffusionNet with L1 loss, and finally WaveDiffusion. Both WaveNet-low and WaveNet-high diverge during training.

4.2 WSS estimation: qualitative results

We investigate visually the performance of the networks trained to estimate the WSS. We reiterate that although each network is trained to predict the WSS vector, i.e. $(\text{WSS}_x, \text{WSS}_y, \text{WSS}_z)$, we compare the magnitude of the estimated signals, since the magnitude is correlated with the components and more concise to interpret.

In Figure 4.1 we show the WSS estimates. Particularly difficult for the networks is the high intensity red area at the bifurcation, where the signal sharply increases in magnitude. DiffusionNet-L2 appears most succesful at estimating this correctly, followed by DiffusionNet with L1 and WaveDiffusionNet. These results are corroborated by the quantitative results table, where these networks have the lowest MAE. Both WaveNet-high and WaveNet-low diverge during training, accumulating train and validation loss. For completeness we show their predictions, but these are incomparably poor.



(a) Ground truth |WSS|

(b) DiffusionNet L1 |WSS| estimate

(c) DiffusionNet L2 |WSS| estimate

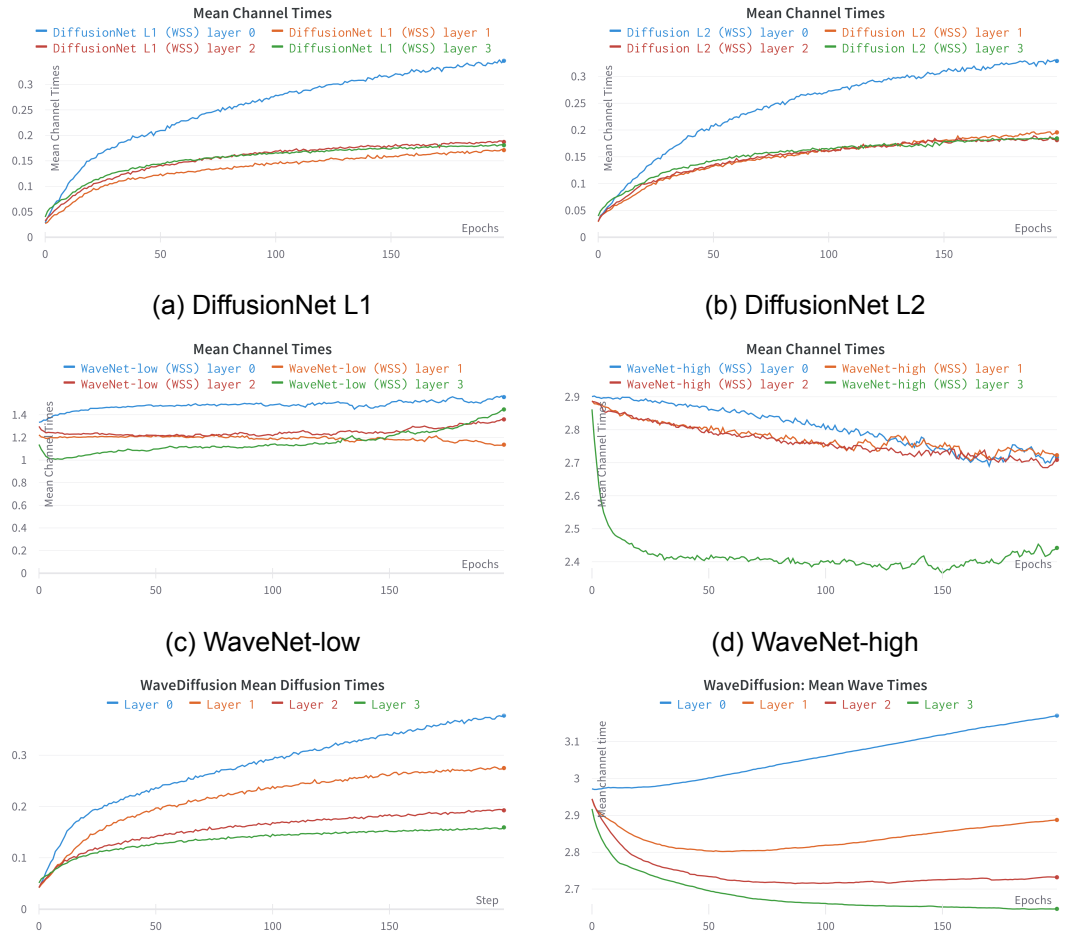
(d) WaveNet-low |WSS| estimate

(e) WaveNet-high |WSS| estimate

(f) WaveDiffusionNet |WSS| estimate

Figure 4.1: Visualization of the magnitude of the estimated WSS signal (|WSS|) by different networks. Particularly difficult for the networks is the high intensity red area at the bifurcation, where the signal sharply increases in magnitude. DiffusionNet-L2 appears most succesful at estimating this correctly, followed by DiffusionNet with L1 and WaveDiffusionNet. These results are corroborated by the quantitative results table, where these networks have the lowest MAE. Both WaveNet-high and WaveNet-low diverge during training, accumulating train and validation loss. For completeness we show their predictions, but these are incomparably poor.

Additionally, we show the layer times for DiffusionNet, WaveNet and WaveDiffusionNet. The layer times of diffusion modules tend to increase from 0, whereas the wave parameters are initialized away from 0 at either 1.5 (WaveNet-low) or 3.0 (WaveNet-high, WaveDiffusionNet). Where the learned diffusion parameters tend to increase monotonously in mean, the learned wave parameters typically show more variability, going up and down. DiffusionNet with L_1 and L_2 loss have similar learned times at epoch 200.



(a) DiffusionNet L1

(b) DiffusionNet L2

(c) WaveNet-low

(d) WaveNet-high

(e) WaveDiffusion diffusion

(f) WaveDiffusion wave

Figure 4.2: Evolution of mean times of each layer within the networks estimating WSS. The mean times are found by averaging the times of all channel times at that layer, and plotted per epoch. In all trained networks, the first layers tend to have the highest mean time at epoch 200.

4.3 Blood Pressure Prediction:

Both WaveNet-low and WaveNet-high have a steeper training loss decrease than DiffusionNet. However, this is not reflected in the validation performance, where all three networks perform similarly. The MAE (Table x) is very low

When looking at the mean of all learned channel time parameters t per epoch, we see in both WaveNet and DiffusionNet a preference for smaller time parameter in early layers. In the last layer we see a striking difference, where the learned time becomes much larger. For diffusion this causes the feature vector to move towards its steady state (uniform) heat distribution. For waves, this means longer-range communication.

BP Networks	NMAE	Approximation Error	Mean Cosine Similarity
WaveDiffusion	0.1	0.1	1.00
DiffusionNet	0.0	0.0	1.00
WaveNet-low	0.0	0.1	1.00
WaveNet-high	0.0	0.0	1.00
PPC	0.0	0.0	1.00

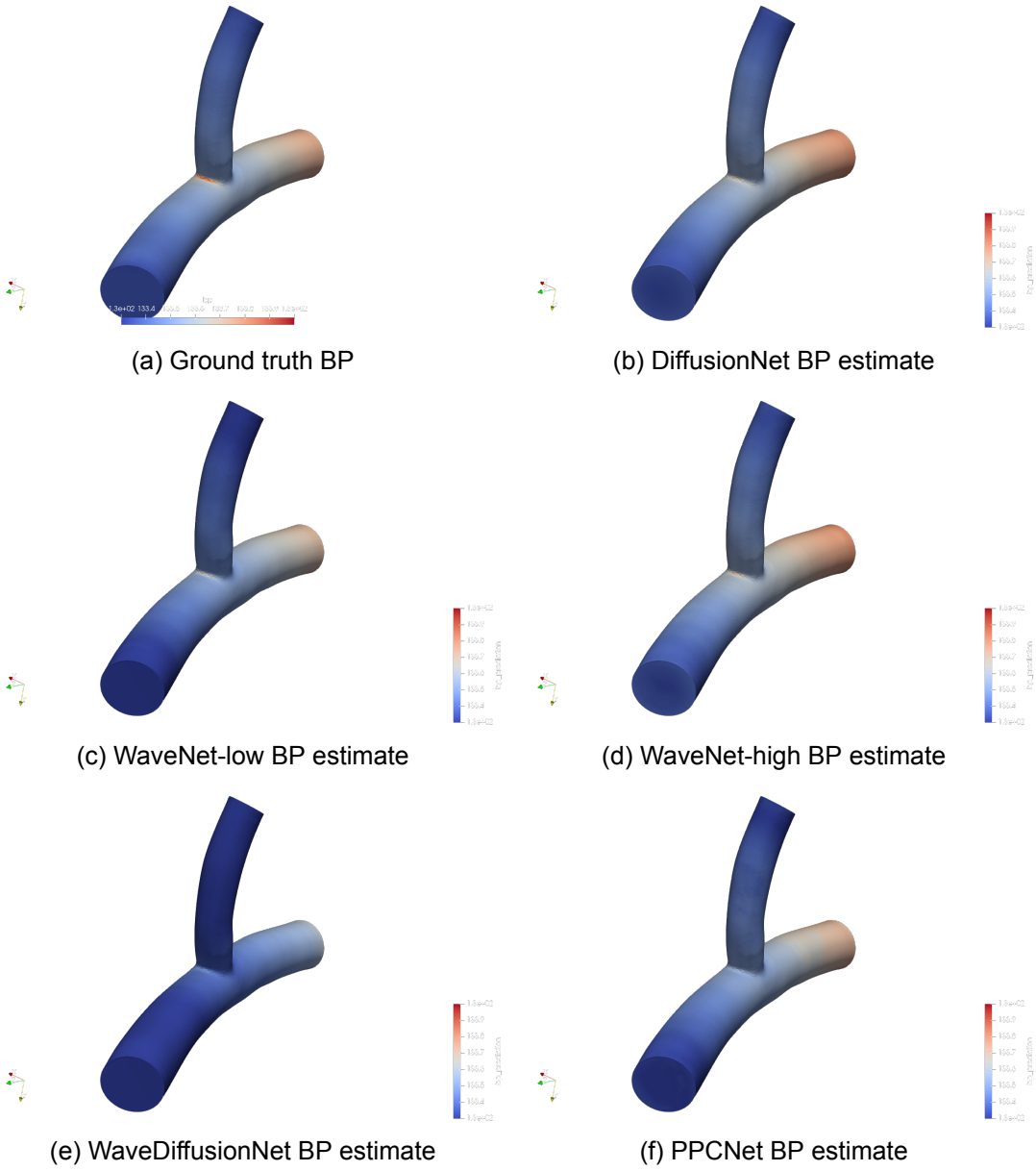
Table 4.2: All networks achieve very low MAE and NMAE on the BP prediction task. This suggests nearly perfect performance. However, as seen in the qualitative evaluation (Figure **??**), this is not the case.

The mean Dirichlet Energy of the channels as they passed through the layers remains fixed, for all of the networks.

Qualitative evaluation The reconstruction of the BP identifies correctly which regions have comparatively lower and higher BP than the mean. However, the reconstruction often undershoots in critical areas, where the ground truth varies rapidly between low and high BP in a small area.

4.4 BP: Qualitative

For our qualitative analysis we investigate the estimated BP signal for all networks trained to estimate the BP. The plots show the signal dynamics were replicated well by DiffusionNet, WaveNet-high and WaveNet-low. These networks appear most succesful at estimating the detail in the BP signal at the bifurcation. In the figure of PPCNet's estimate, we see a low signal magnitude at the bifurcation. Thus, despite having high quantitative performance, the visual reconstruction of the signal misses detail.



(a) Ground truth BP

(b) DiffusionNet BP estimate

(c) WaveNet-low BP estimate

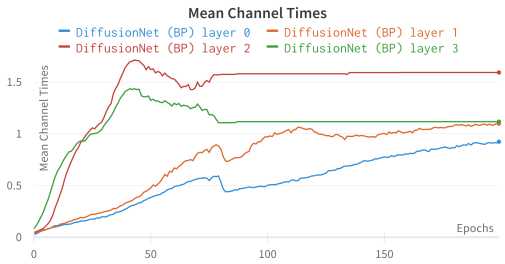
(d) WaveNet-high BP estimate

(e) WaveDiffusionNet BP estimate

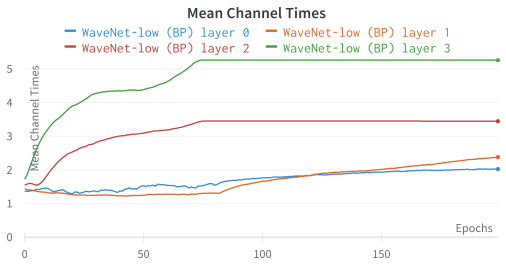
(f) PPCNet BP estimate

Figure 4.3: Visualization of estimated BP signal by different networks. Particularly difficult for the networks is the high intensity red area at the bifurcation, where the signal sharply increases in magnitude. DiffusionNet, WaveNet-high and WaveNet-low appear most succesful at estimating this detail in the BP signal. In the figure of PPCNet's estimate, we see a low signal magnitude at the bifurcation. Thus, despite having high quantitative performance, the visual reconstruction of the signal misses detail.

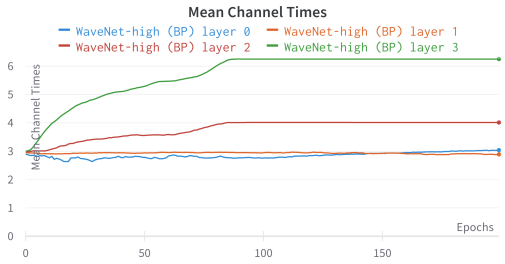
Additionally we show the mean times of each layer for the BP estimation task.



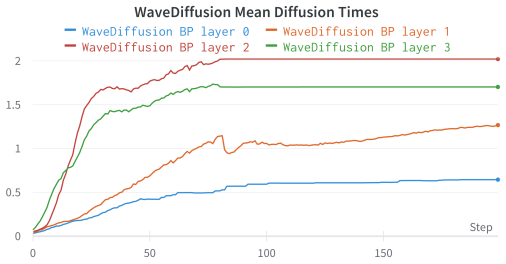
(a) DiffusionNet



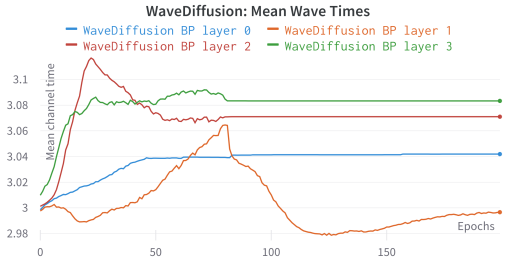
(b) WaveNet-low



(c) WaveNet-high



(d) WaveDiffusion diffusion



(e) WaveDiffusion wave

Figure 4.4: Evolution of mean times of each layer within the networks estimating BP. The mean times are found by averaging the times of all channel times at that layer, and plotted per epoch. After a certain point, most layers stabilize in the BP prediction task. In all trained networks, the last layers tend to have the highest mean time at epoch 200.

4.5 Frequency reconstruction experiments

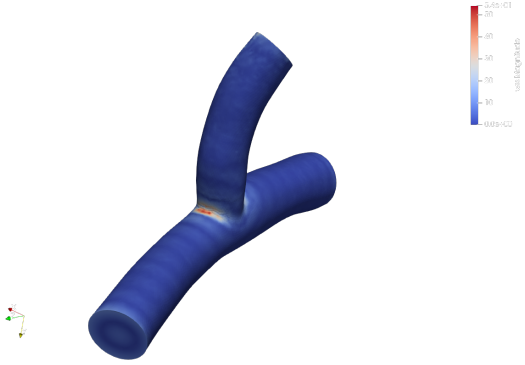
On the 50 mesh dataset for estimating high-pass filtered WSS, WaveNet outperforms DiffusionNet.

Reconstruction	NMAE	Approximation Error	Mean Cosine Similarity
WaveNet	0.6 ± 0.1	26.6 ± 3.1	0.91 $\pm$ 0.02
DiffusionNet	1.4 - 0.4	45.5 - 5.6	0.58-0.06

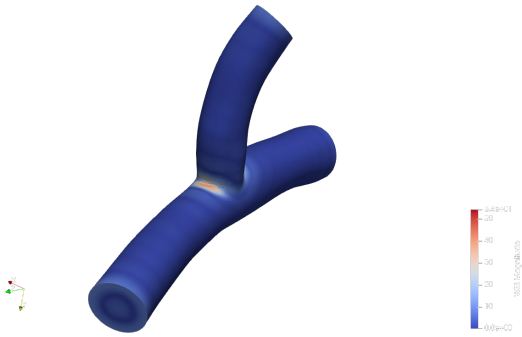
Table 4.3: WSS reconstruction metrics for the overfitting experiment on the dataset of 50 highpass filtered train samples. Though neither network attains a low error, WaveNet outperforms DiffusionNet by a large margin.

In Figure 4.5 we see the WSS signal filtered to remove only the first 178 components. Despite both networks significantly underestimating the signal strength, especially near the bifurcation, WaveNet-high estimates appears to estimate the magnitude of the WSS more accurately than

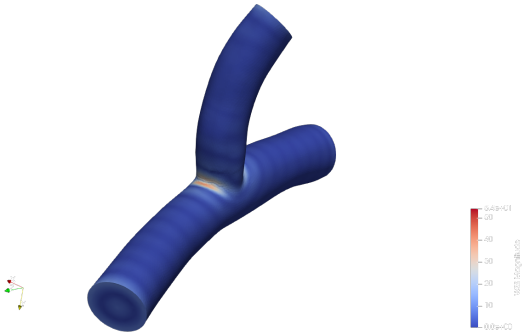
DiffusionNet.



(a) Ground truth high-pass filtered WSS (first 178 eigenbasis components removed)



(b) DiffusionNet estimation of high-pass filtered WSS

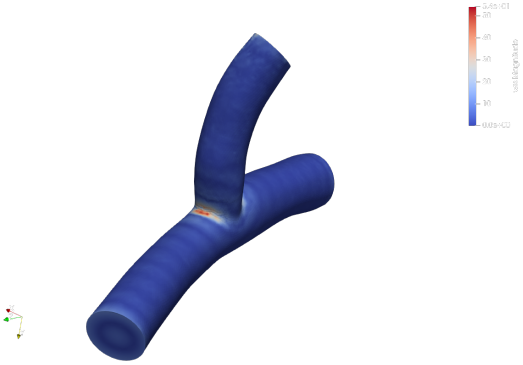


(c) WaveNet-high estimation of high-pass filtered WSS

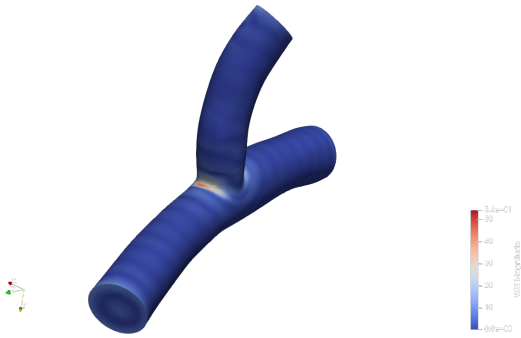
Figure 4.5: Visualizations of estimated high-pass filtered WSS for DiffusionNet and WaveNet-high. Both networks estimate correctly the structure of the signal, and which areas should be relatively low or high magnitude. However, both networks underestimate the magnitude. Visually, the representations look nearly identical, though WaveNet's estimate appears to have a slightly higher magnitude near the bifurcation. Additionally, DiffusionNet is unable to capture the wave-like signal variation of the ground truth, producing an estimate that is a lot smoother.

Reconstruction	NMAE	Approximation Error	Mean Cosine Similarity
WaveNet	0.9	22.1	0.89
DiffusionNet	1.0	25.6	0.88

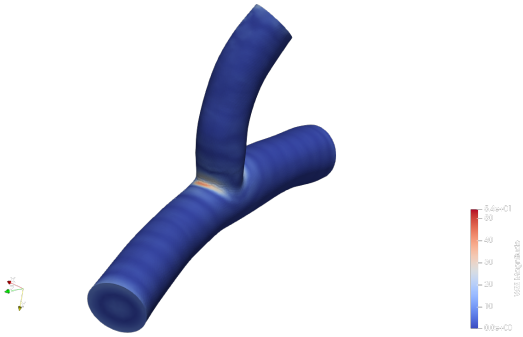
Table 4.4: DiffusionNet and WaveNet attain similar performance in learning to estimate the WSS on one mesh.



(a) Ground truth high-pass filtered WSS (first 178 eigenbasis components removed)



(b) DiffusionNet estimation of high-pass filtered WSS



(c) WaveNet-high estimation of high-pass filtered WSS

Figure 4.6: Visualizations of estimated high-pass filtered WSS for DiffusionNet and WaveNet-high on one mesh. Both networks again estimate correctly the structure of the signal, and which areas should be relatively low or high magnitude. Visually, the representations look nearly identical.

4.6 Oversmoothing and Dirichlet energy

DiffusionNet did not seem to have lower Dirichlet energy than wave-based networks. Furthermore, the Dirichlet energy of DiffusionNet did not appear to go down, rather it went up, as the signal passed through the layers.

WSS Networks	After linear layer	Layer 0	Layer 1	Layer 2	Layer 3
WaveDiffusion	350	800	2,092	4,043	4,558
DiffusionNet L1	333	1,007	2,718	4,736	5,095
DiffusionNet L2	307	1,033	4,125	5,985	6,341
WaveNet-low	253	61,192	150,287	244,793	314,139
WaveNet-high	218	26,705	75,836	134,509	162,239
PPC	262	185,312	250,005	533,926	610,894

Table 4.5: Dirichlet energy mean over all channels after the linear layer and at each subsequent network layer, for networks trained to estimate the WSS. For each network, the energy of the signal increases as it moves through the network. Notably, both WaveNets have incredibly high energy. Values are rounded to the nearest whole number.

BP Networks	After linear layer	Layer 0	Layer 1	Layer 2	Layer 3
WaveDiffusion	93.018	93.018	92.927	92.927	92.927
DiffusionNet	100.586	100.81	100.924	100.924	100.924
WaveNet-low	59.539	61.33	62.035	62.035	62.035
WaveNet-high	52.647	52.907	52.907	52.907	52.907
PPC	100.702	100.913	100.913	100.913	100.913

Table 4.6: Dirichlet energy mean over all channels after the linear layer and at each subsequent network layer, for networks trained to estimate the BP. For each network, the energy of the signal increases as it moves through the network. The energy does not undergo a large change in magnitude after the linear layer.

5 DISCUSSION

In this section we interpret our results, and discuss the limitations of the wave kernel we use specifically, and our methods generally. We discuss the answer to our research question, and to what extent the limitations prevent us from answering it. We also provide some directions for future research.

5.1 Interpretation of Results

DiffusionNet can learn signals on arterial meshes to a high degree of accuracy, as evidenced by its close performance to GEM-CNN. However, some of the detail in the reconstructions appeared to be missing, and the test performance was slightly lower than the state of the art, GEM-CNN. Equipping DiffusionNet with an L_2 loss function appeared to increase test accuracy.

Networks using the wave equation were not able to compete well with DiffusionNet, including WaveDiffusion, which should theoretically have a more expressive kernel than DiffusionNet by being a combination of diffusion and wave dynamics.

Although we cannot rule out potential other numerical schemes, we believe the spectral wave equation within the framework of DiffusionNet does not give the network enough freedom to learn the right feature dynamics for WSS prediction.

The PPC networks achieved poor performance, despite having more parameters and access to more information about the signal via the different PDE channels. We expect this was caused by the instability from training with the wave kernel, and potentially also from a lack of normalization since so many channels were added in the PPC layers. However, our attempts to remedy PPCNet's performance issues via BatchNorm were unsuccesful.

We also showed that for smooth signals, the wave kernel propagates them in a wave-like manner. We investigated the behaviour of the wave kernel via point sources, to get a better understanding of how vertices affect each other as the feature evolves over the surface, and find evidence for wave-like behavior.

For the blood pressure prediction, our quantitative metrics were not suitable for a signal concentrated in magnitude around $(\mu - \epsilon, \mu + \epsilon)$ where the mean of the signal $\mu >> \epsilon$. In other words, the detail resides in a very small scale, making MAE and NMAE insufficient metrics. A better approach would have been to center the BP signal around a normal distribution before training the network, by reshaping it for each artery as

$$\mathrm{BP}_{\mathrm{normalized}} = \frac{\mathrm{BP} - \mu}{\sigma^2} \tag{5.1}$$

with μ and σ the mean and standard deviation of the blood pressure for that artery.

It appears that even though the BP is a global feature, both waves and diffusion can approximate it equally well. This implies specific long-distance interactions may be less impactful than previously thought, and the prediction of BP may simply require a feature vector containing global information, which diffusion for large time does produce. Alternatively, it is possible our wave based networks were not able to establish long-range feature communication.

WSS high-pass filter reconstruction task: despite poor performance in the WSS task, WaveNet was able to beat DiffusionNet in the reconstruction tasks for overfitting onto a small dataset. This points to WaveNet potentially learning high frequency content better. However, since it was a small dataset and the task was specifically to overfit onto the dataset, this does not necessarily point to good performance on the test set.

Dirichlet energy: the Dirichlet energy did not change a significant amount per layer for the BP prediction. A possible explanation is that the hidden layer signal did not change significantly either, and the network perhaps didn't need this many layers.

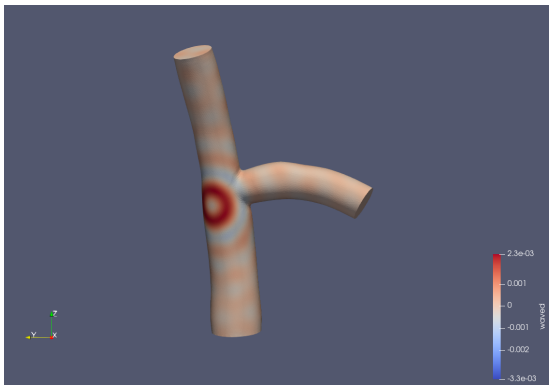
For the WSS we see the signal Dirichlet energy increase a lot. From these results, it does not look like adding extra network layers would cause oversmoothing. This is something that could be investigated in the future.

5.2 Limitations of the Spectral Wave Kernel

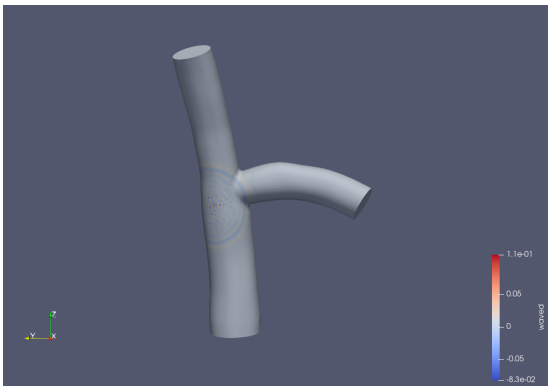
Both the spectral wave kernel, and our attempts to expose its properties, have some limitations. We split these shortcomings into numerical and conceptual. Additionally, we discuss some gaps in our understanding of the kernel's behaviour, and how we could solve those where applicable.

5.2.1 Numerical

First, when t is large, we observed the point source dynamics get jittery and irregular, suggesting the wave behaviour might only be present for early times. However, a continuous evolution of a feature vector did not suffer from this behaviour, suggesting the numerical inaccuracies may not be significant on a large scale. Similarly, the point source image (Figure 5.1) with reduced eigenbasis shows multiple wavefronts coming from the source node. In contrast, the spectral wave equation evaluated using an almost complete (17,000 out of around 19,000 vectors) eigenbasis there was only one wavefront coming from the source node, but with irregularities at the center. While the wavefronts in the image with 256 components could point to additional numerical inaccuracies caused by projecting to a reduced basis, the full basis appears to have irregularities at the center. We cannot rule out that these irregularities result from not having access to the remaning 2000 basis vectors. However, neither wave propagates as expected.



(a) Point source at $t = t_0$ under wave dynamics using eigenbasis with 256 components.

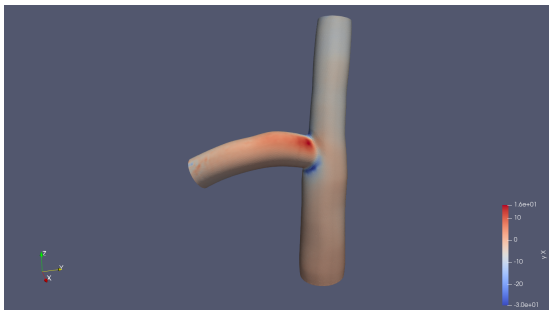


(b) Point source at $t = t_0$ under wave dynamics using (nearly) full eigenbasis.

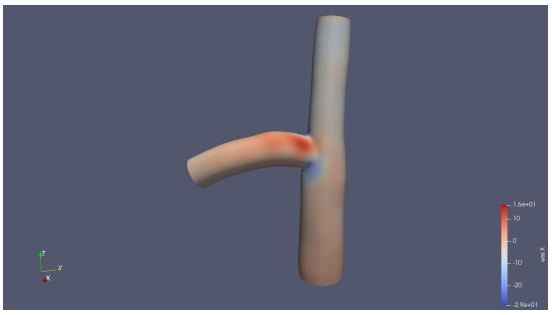
Figure 5.1: The above evolution of the point source on the right was calculated with an eigenbasis of 17,000 eigenvectors, on the left with 256. We see that, in comparison to the reduced basis, the wave contains a lot less oscillations, and the displacement is strictly localized around the center of the point source. Additionally, the wave with full basis has some irregularities at the center where certain nodes have a high value, contrary to what we would expect.

Secondly, the spectral benefits of the wave kernel may be limited. Although the kernel itself does not attenuate these basis vectors as much as DiffusionNet, some are still removed. It is also good to check the coefficients generated by DiffusionNet, $e^{-\lambda_i t}$ at each layer for each t, as diffusion for a very small time does not remove as much high frequency information as for large time. Investigating this would give us a better understanding of whether the low-pass property of DiffusionNet is truly limiting it or not.

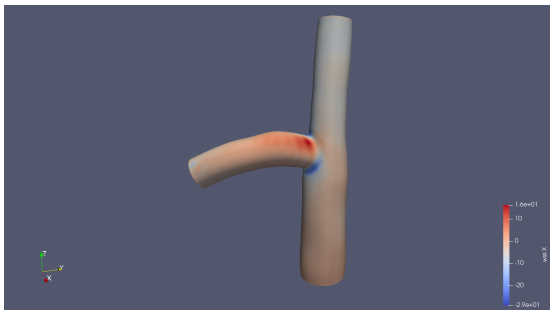
Additionally, it is unclear to what extent the basis vectors used in WaveNet (the first 256) contribute to detail. A reconstruction of the WSS using only those eigenvectors was rather blurry. It is possible that having access to these higher frequencies can still help WaveNet learn more complicated signals. We could investigate how the spectrum of the hidden layer signal changes as it goes through the layers, to verify whether the models are using high-frequency basis vectors in the hidden layer signals.



(a) Ground truth WSS x-component



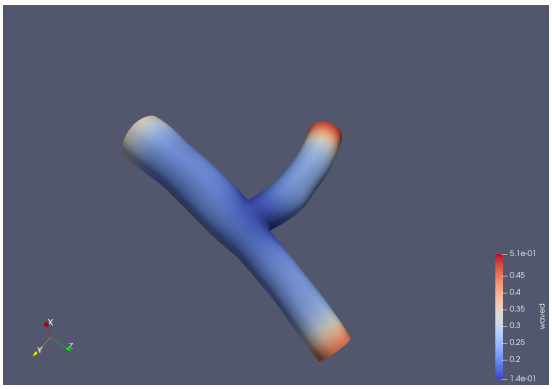
(b) First 256 eigenvectors



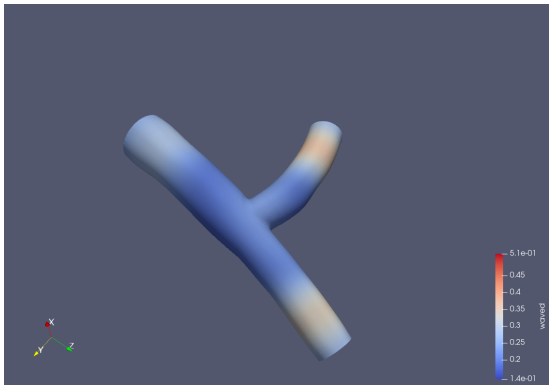
(c) First 1700 eigenvectors

Figure 5.2: The x-component of the WSS is projected onto spectral bases of different number of eigenvectors. The signal, expressed in the reduced bases, is then projected back to the vertex domain. We see the jittery nature of the WSS signal is most present in the reconstruction using the higher bases.

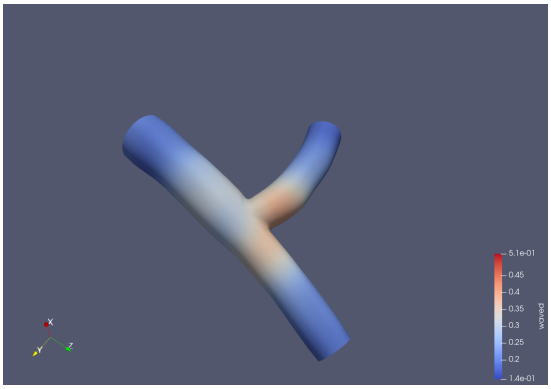
We use point sources to help explain the wave dynamics. However, the point source in Figure 5.1 (a) shows multiple wavefronts coming from the source node. Both negatively and positively weighted nodes play into the current nodes current update. It is hard to determine the magnitude of the oscillations, but the sparseness we see and aim for in the point source with full eigenbasis, is not achieved by the reduced eigenbasis. In future visualizations of point sources, we believe plotting per vertex the absolute value of the wave propagated point source is more descriptive, since that tells us the magnitude of the effect on that vertex.



(a) Initial feature distribution



(b) Feature after propagating the wave equation for 10 timesteps



(c) Feature after propagating the wave equation for 26 timesteps

Figure 5.3: An initial feature (a) is propagated for different amounts of time. The amplitude of the feature is conserved from (b) to (c). The wave equation provides a way to propagate features over the mesh, to distant nodes. This shows the wave equation should be a good candidate for long-range message passing over a manifold. The wave equation is evaluated using the spectral wave kernel [18].

We also propagate signals input features on the mesh (Figure 5.3) to verify the wave kernel's behaviour. Since these signals were themselves rather smooth, they may not have suffered as much from the reduced eigenbasis. It is likely the wave kernel still has smoothing properties due to its reduced eigenbasis, so high frequency components in the signal will still be attenuated because they are projected to and from a reduced basis. This could be tested by propagating signals containing more high frequency content and investigating visually whether the wave dynamics are still present.

Finally, the wave kernel $g(\lambda_l, t) = \cos(t \cos^{-1}(1 - \frac{s\lambda_l}{2})$ is defined relative to the maximum eigenvalue of the mesh, because we must choose $s < \frac{4}{\lambda_{\text{max}}}$ [18]. In our case, since we did not compute the full spectrum for each mesh, we used $s = \frac{2}{\lambda_{\text{max}}}$ since $\lambda > 0$, giving us unconditional stability. However, this means we do have varying wave speed across meshes, proportional to their largest eigenvalue in the decomposition. While we do not expect this to have a big impact on performance because the max eigenvalues are likely similar across different meshes, this should be shown empirically or theoretically.

5.2.2 Conceptual

The point source interpretation of the spectral wave kernel may not be directly related to WaveNet. Instead of showing the properties of the kernel visually via the evolution of a point source (Figure

5.1), we should investigate the properties of the wave operator matrix $H(t) = V^T g(\lambda, t)V$. While the point source effectively takes one row from $H(t)$ and therefore should show how the point source influences the vertices around it, it would be good to examine $H(t)$ more rigorously, to determine the connection between propagating the point source and propagating a whole feature vector. A smaller mesh might make this easier, since the amount of vertex interactions is minimized. If we improve our understanding of how the signal in the hidden layers is updated via the PDE operator, we might be able to relate the DiffusionNet structure to existing network structures such as Graph Attention, MPNN, and transformers.

Networks using the wave kernel are prone to divergence during training, in both train and validation/test loss. This could be a result of the learning rate being too high, but it could also mean the wave kernel is not suitable for learning on surfaces within the DiffusionNet neural network architecture. This is because DiffusionNet evolves the feature vector globally by a time t, which makes long-distance interactions difficult to define since they are shared by all nodes.

Since the wave kernel is isotropic, a theoretical verification of whether the spatial gradient features of DiffusionNet still apply would also be desirable.

This brings us to another limitation: the wave dynamics are the same on the whole surface. Hypothetically, if the wave kernel allowed long-distance communication, relying on this in a layer would be difficult, since the distance between vertices communicating is fixed over the entire mesh due to a global learnt time t. A possible extension could be to instead solve the anisotropic wave equation, similar to how [5] solves the anisotropic heat equation. By letting features propagate at different speeds in different parts of the mesh, we might allow the network to express more complex interactions. However, the anisotropic heat equation is already difficult to solve for non time-varying coefficients, and in [5] a special anisotropic Laplacian is defined in order to solve it.

Also, in [45] it is mentioned that linear spectral networks can only reconstruct signal frequencies that are present in their input. Although our network is not a linear spectral network, meaning high-frequency information can be added to the signal via the MLP, we still project our input signal onto a spectral basis and scale it with a spectral filter. Especially for the wave network, it could be that the absence of high frequency input features causes the waves to have minimal impact.

The spectral wave kernel also requires computing the eigenbasis of the cotangent Laplacian in advance. This presents two problems. The first is the choice of basis size. For diffusion, since it is a low-pass filter the choice is easier, but since the wave equation is more like a band-pass filter, it is difficult to choose how many eigenvectors the spectral kernel should have access to. Secondly, precomputing the basis can be time consuming, and especially on surfaces which only need to be evaluated once, may be a lot slower than other methods. Since the basis is intrinsic to the surface, this is well worth it when evaluating multiple passes of the network on the same surface.

It is currently unclear how diffusion to time t_0 translates to a larger mesh, and whether this still encompasses the same area as on a smaller mesh, or if the area is relative to the total area of the shape. This could be important in the future when training models operating on manifolds that are of varying spatial dimension.

The arteries used to investigate the wave dynamics were reasonably simple in shape, without many branches or thinner parts. It is unclear to what extent the wave dynamics are present for

larger, more complicated structures.

The above limitations of our method make it difficult to precisely determine which part of the method caused poor results. Because propagating a feature over the mesh in aggregate seemed to yield wave-like behaviors, we estimate the numerical properties cannot be the sole cause of the performance issues. The most impactful cause of the low performance seems to be the fact that learning a single time parameter for the whole manifold at each layer is too general, and goes against the nature of long-distance interactions.

Despite these limitations, we believe the results confirm that the spectral wave kernel is not able to learn long distance feature interactions in a way that meaningfully helps the network predict the WSS.

Additionally, since the kernel was initially developed for and used on a graph, it should be investigated whether the kernel works as expected on edges and corners such as the inlet, where the mesh suddenly bends.

It is currently unclear how the spectral wave equation behaves on more complicated meshes. When the surface is more complicated, we might need a larger portion of the Laplacian spectrum to obtain an accurate solution to our PDEs.

5.3 General limitations of our methods

For all trained models the validation performance varied even as the network appeared to have converged in train loss, and the standard deviation of the errors was quite large for all models as well. Thus, despite approximating the signal to a reasonable degree, the networks were still inconsistent.

Evaluation metrics: the AE and MCS proved insightful metrics, however, it may be worth investigating whether a metric more discriminative to outliers could help interpret signals. Visually, it is virtually impossible to compare the WSS estimations of different networks. Often the estimates of different networks look similar, and the interpreter of the visualizations provides only a subjective answer on whether key areas have been correctly estimated. Therefore, a quantitative measure specifically tailored to regression of spatially variant signals on a mesh based domain could be useful.

Another limitation of the MAE is the fact it is node based, whereas we wish to estimate the reconstruction on the surface. Therefore, the error should factor in the area of the faces via the mass matrix M of the mesh, such as $\text{MAE}_{\text{new}}(u) = \text{MAE}(M \cdot u)$. In [22] and [1] some detail is given on using Mass matrices on meshes to define energy norms.

Similarly, we currently output our signal at the mesh vertices. However, since we aim to predict a signal of the underlying manifold and not the mesh, we might consider estimating the signal on the mesh faces instead.

There is no bias toward estimating specific nodes or areas correctly. Because the WSS is spatially variant over only a small number of nodes, the MAE does not provide a clear answer to whether the network was able to estimate this information in the signal. Alternatively, an error with a bias to certain high variance areas could be constructed. We suggest potentially using

[1] Link to post

the spatial gradient of the signal on the surface as a way to weight the loss function toward being more mindful of areas with high spatial variance.

Our Dirichlet energy calculation did not factor in local mesh resolution. Instead of DE $= u^T L u$, an alternative formulation should use the mass matrix to incorporate mesh areas. However, we expect this does not greatly change the upward trend seen in the energy per layer of the networks.

Crucial to the success of the Laplace operator is that it represents the underlying surface and not the particular mesh discretization. Since DiffusionNet generalizes well to different meshes [35] and they use the cotangent Laplacian, we assume this implies the Laplace operator is reasonably stable under remeshing. However, we could investigate this more formally.

5.4 Future directions

Different PDEs: the reason we employ PDE's is to regularize vertex interactions via a process that occurs on the manifold, in this case the vessel surface rather than the mesh discretization. We can explore different PDE's, such as convection diffusion, or PDE's with potential terms such as the Hamiltonian. Similar to the WKS, we may be able to define our dynamics based on where such a particle might be found.

Alternatively, we could use a random process to define our vertex feature communications. This could be done via random walks such as done in [23], only with probabilities regularized by a physical process, ensuring we select a vertex according to some probability distribution. This avoids computational issues such as computing eigenbases or solving large PDEs. Random processes are also linked to solving PDEs via the Feynman-Kac theorem, suggesting an interesting theoretical link. For example, the Feynman-Kac theorem is used in [10] to solve their diffusion process with a potential. Similar methods might be available for a variety of (parabolic) PDEs.

Rotational Equivariance: for many signals it is important to have equivariant behaviour with respect to rotation. This means, for example, if we predict the WSS and then rotate the predicted mesh, this gives the same result as first rotating the mesh and input features, and then predicting the WSS.

For DiffusionNet this equivariance is not present in the network. In fact, by design the network is invariant via its use of the dot-product in the spatial gradient features, and the fact that diffusion is a process intrinsic to the surface [35].

Although we can achieve some level of rotational equivariance through data augmentation, i.e. randomly rotating our training data and label, a theoretical approach such as used by [19] could be helpful.

Alternative PDE kernels: the kernel we use, found by [18], creates dynamics that resemble waves based on the hyperbolic wave equation. However, the difference between it and the Schrödinger propagator $e^{i\Delta}$, which solves the Schrödinger wave equation, could also be investigated to see if the latter has desirable properties. A starting point could be adapting the method from [1], which also solves the Schrödinger wave equation.

Choice of Discrete Laplacian: the cotangent Laplacian is one scheme of many that exist to

model the Laplacian. In [48] an elaborate treatment is given. For manifold triangle meshes the cotangent Laplacian behaves well and is generally used. If the mesh is non-manifold or not a triangle mesh, a different discretization may need to be used.

Coarse-to-fine method while DiffusionNet is able to estimate the overall structure of the signal, it is unable to reconstruct detail well. Similar to techniques in point-cloud registration, a coarse-to-fine network architecture could be investigated, with DiffusionNet providing a robust estimate, and another network skilled at learning detail could complete the last step. Using wave dynamics for this network could work, since we saw WaveNet beat DiffusionNet purely on reconstruction of high-frequency mesh signals. DiffusionNet could be used for a coarse WSS estimate and WaveNet for refinement.

Batching: our current implementation does not support batching for either DiffusionNet or WaveNet. Implementing this would allow us to train more quickly at the expense of using more memory. This tradeoff is made more favorable due to the fact that DiffusionNet is a small model.

Uncertainty Prediction: the use of Machine Learning methods in medical setting must be supported by a rigorous assessment of the uncertainty of the prediction. Commonly, dropout layers can be used to provide a probabilistic estimate of the network's uncertainty. Dropout layers are layers within the network where connections have a chance p, specified per layer, to be dropped: the weight on that layer will effectively become zero and no backpropagation happens through that path. This prevents the network from overfitting onto certain connections within the MLP. These dropout layers can be removed during testing by setting $p = 0$, however, we can also keep the dropout layer. By passing the same sample through the network multiple times, the dropout layers will ensure we get different predictions. Through a process called bootstrapping, we can estimate the bias and variance of the predictions for this sample. If all the predictions are comparable, we can reason that the network is certain of its prediction. However, if passing the same sample through the network twice results in dramatically different predictions, we might decide not to trust the network's estimation [16].

5.5 Operator Bases: Beyond the Laplace Operator

Hamiltonian: a spectral basis may also be defined using the Hamiltonian instead of the Laplacian Beltrami Operator. The Hamiltonian allows us to use a potential function, defined on each vertex of the mesh, which affects the way a signal diffuses. We could define a potential favoring for example diffusion into areas near the bifurcation, to accumulate features there. [10]

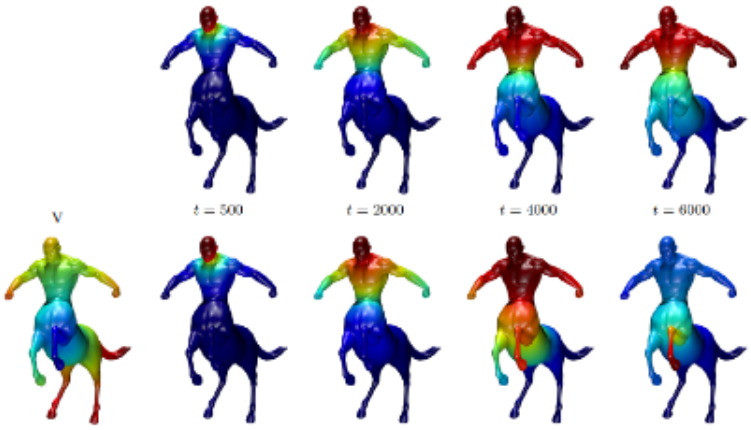


Figure 5.4: This figure, taken from [10], shows the process of spectral diffusion using the basis from the Laplace Beltrami Operator, versus the Hamiltonian basis.

As seen in Figure 5.4, a diffusion process with a potential defined as the geodesic distance to the lower leg diffuses anisotropically to regions with low potential.
An example application within coronary arterial meshes could be to define our potential function as the geodesic distance from the inlet, which would make all the features diffuse toward those the inlet. Such a method would provide a transferable approach to any mesh geometry, while remaining a geometrically motivated feature communication mechanism.

Steklov Operator: Intrinsic operators such as the Laplacian lack the ability to differentiate between the cube with an inward bump and the cube with an outward bump. The Steklov Operator is an extrinsic geometry operator. This operator is aware of the volume confined within the surface, making it a good choice for arterial models. The authors describe natural generalizations of PDE's, HKS and WKS using it. [47][46]

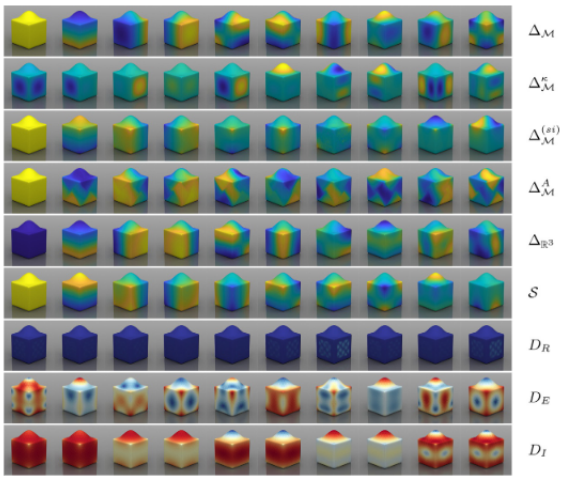


FIG. 9 First few eigenfunctions computed on the cube with an outward bump. For the Dirac operators, norms of the (quaternion valued) eigenfunctions are visualized.

Figure 5.5: Eigenfunctions of different operators on the cube with outward bump. Figure taken from [46]

6 CONCLUSION

DiffusionNet [35] generalizes well to arterial meshes and predicting complicated signals on them. The robustness to discretization makes this a promising network architecture in medical deep learning on surfaces.

The spectral wave kernel was unable to cause significant performance increase in the trained networks for WSS or BP estimation. In some cases, the kernel appeared to cause instability during training.

Though the spectral wave kernel appears to cause the feature to propagate in a wave-like manner, this is not a conclusive result. Additionally, it is unclear whether the wave kernel was unable to create long-range interaction, or whether only being permitted to learn time t for the whole channel was a limiting factor for the kernel within WaveNet.

Finally, the wave kernel did show promise in the overfitting task, where networks overfit onto a small subset of the training data. Estimating the high-pass WSS showed a clear performance gap between WaveNet and DiffusionNet. This could hint at increased accessibility to higher frequencies in WaveNet helping it reconstruct detail.

Bibliography

[1] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. “The wave kernel signature: A quantum mechanical approach to shape analysis”. In: *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE. 2011, pp. 1626–1633.

[2] Maysam Behmanesh, Maximilian Krahn, and Maks Ovsjanikov. “TIDE: Time Derivative Diffusion for Deep Learning on Graphs”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 2015–2030.

[3] J. Benartzi. *Linear Wave Equations*. Last accessed 26 June 2023. url: https://jbenartzi.github.io/2015.Dispersive/files/5.pdf.

[4] UC Berkley. *Adjoint and self-adjoint operators and matrices*. 2015. url: %5Curl%7Bhttps://inst.eecs.berkeley.edu/~ee16b/su20/lecture/lec11.pdf%7D.

[5] Davide Boscaini et al. “Learning shape correspondence with anisotropic convolutional neural networks”. In: *CoRR* abs/1605.06437 (2016). arXiv: 1605.06437. url: http://arxiv.org/abs/1605.06437.

[6] Michael Bronstein. “Beyond Message Passing: a Physics-Inspired Paradigm for Graph Neural Networks”. In: *The Gradient* (2022).

[7] Michael M Bronstein et al. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[8] Michael M. Bronstein et al. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges”. In: *CoRR* abs/2104.13478 (2021). arXiv: 2104.13478. url: https://arxiv.org/abs/2104.13478.

[9] Joan Bruna et al. “Spectral networks and locally connected networks on graphs”. In: *arXiv preprint arXiv:1312.6203* (2013).

[10] Yoni Choukroun et al. “Hamiltonian operator for spectral shape analysis”. In: *IEEE transactions on visualization and computer graphics* 26.2 (2018), pp. 1320–1331.

[11] Keenan Crane. “Discrete differential geometry: An applied introduction”. In: *Notices of the AMS, Communication* 1153 (2018).

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems* 29 (2016).

[13] Francesco Di Giovanni et al. “On over-squashing in message passing neural networks: The impact of width, depth, and topology”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 7865–7885.

[14] Moshe Eliasof, Eldad Haber, and Eran Treister. “PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations”. In: *CoRR* abs/2108.01938 (2021). arXiv: 2108.01938. url: https://arxiv.org/abs/2108.01938.

[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.

[16] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[17] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG].

[18] Francesco Grassi, Nathanaël Perraudin, and Benjamin Ricaud. “Tracking time-vertex propagation using dynamic graph wavelets”. In: *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2016, pp. 351–355.

[19] Pim De Haan et al. “Gauge Equivariant Mesh {CNN}s: Anisotropic convolutions on geometric graphs”. In: *International Conference on Learning Representations*. 2021. url: https://openreview.net/forum?id=Jnspzp-oIZE.

[20] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. “Wavelets on graphs via spectral graph theory”. In: *Applied and Computational Harmonic Analysis* 30.2 (2011), pp. 129–150.

[21] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[22] Alec Jacobson. *Understanding mass matrix lumping in terms of functions spaces*. https://www.alecjacobson.com/weblog/?p=4666.

[23] Alon Lahav and Ayellet Tal. “MeshWalker: Deep Mesh Understanding by Random Walks”. In: *CoRR* abs/2006.05353 (2020). arXiv: 2006.05353. url: https://arxiv.org/abs/2006.05353.

[24] Jonathan Masci et al. “Geodesic convolutional neural networks on riemannian manifolds”. In: *Proceedings of the IEEE international conference on computer vision workshops*. 2015, pp. 37–45.

[25] Mathworks. *Fourier Transform*. https://nl.mathworks.com/help/images/fourier-transform.html. Last accessed 26 June 2023. 2017.

[26] Federico Monti et al. “Geometric deep learning on graphs and manifolds using mixture model CNNs”. In: *CoRR* abs/1611.08402 (2016). arXiv: 1611.08402. url: http://arxiv.org/abs/1611.08402.

[27] Kenta Oono and Taiji Suzuki. “Graph neural networks exponentially lose expressive power for node classification”. In: *arXiv preprint arXiv:1905.10947* (2019).

[28] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[29] Ulrich Pinkall and Konrad Polthier. “Computing discrete minimal surfaces and their conjugates”. In: *Experimental mathematics* 2.1 (1993), pp. 15–36.

[30] Adrien Poulenard and Maks Ovsjanikov. “Multi-directional geodesic neural networks via equivariant convolution”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–14.

[31] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. “A survey on oversmoothing in graph neural networks”. In: *arXiv preprint arXiv:2303.10993* (2023).

[32] T Konstantin Rusch et al. “Graph-coupled oscillator networks”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 18888–18909.

[33] Lars Ruthotto and Eldad Haber. “Deep Neural Networks motivated by Partial Differential Equations”. In: *CoRR* abs/1804.04272 (2018). arXiv: 1804.04272. url: http://arxiv.org/abs/1804.04272.

[34] Aliaksei Sandryhaila and José MF Moura. “Discrete signal processing on graphs: Graph fourier transform”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 6167–6170.

[35] Nicholas Sharp et al. “Diffusion is All You Need for Learning on Surfaces”. In: *CoRR* abs/2012.00888 (2020). arXiv: 2012.00888. url: https://arxiv.org/abs/2012.00888.

[36] Justin Solomon. *Discrete Laplacian Operators*. https://groups.csail.mit.edu/gdpgroup/assets/6838_spring_2021/9_discrete_laplacians.pdf. 2021.

[37] David Stutz. *Learning Shape Completion from Bounding Boxes with CAD Shape Priors*. http://davidstutz.de/. Aachen, Germany, Sept. 2017.

[38] Julian Suk et al. “Equivariant graph neural networks as surrogate for computational fluid dynamics in 3D artery models”. In: *Proceedings of the Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS)*. 2021.

[39] Julian Suk et al. *Mesh Neural Networks for SE(3)-Equivariant Hemodynamics Estimation on the Artery Wall*. 2022. arXiv: 2212.05023 [cs.LG].

[40] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. “A concise and provably informative multi-scale signature based on heat diffusion”. In: *Computer graphics forum*. Vol. 28. 5. Wiley Online Library. 2009, pp. 1383–1392.

[41] Jake Topping et al. “Understanding over-squashing and bottlenecks on graphs via curvature”. In: *arXiv preprint arXiv:2111.14522* (2021).

[42] Pim Van Ooij et al. “Aortic valve stenosis alters expression of regional aortic wall shear stress: New insights from a 4-dimensional flow magnetic resonance imaging study of 571 subjects”. In: *Journal of the American Heart Association* 6.9 (2017), e005959.

[43] Petar Veličković et al. “Graph Attention Networks”. In: *International Conference on Learning Representations* (2018). url: https://openreview.net/forum?id=rJXMpikCZ.

[44] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. doi: 10.1038/s41592-019-0686-2.

[45] Xiyuan Wang and Muhan Zhang. “How powerful are spectral graph neural networks”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23341–23362.

[46] Yu Wang and Justin Solomon. “Intrinsic and extrinsic operators for shape analysis”. In: *Handbook of Numerical Analysis*. Vol. 20. Elsevier, 2019, pp. 41–115.

[47] Yu Wang et al. “Steklov Geometry Processing: An Extrinsic Approach to Spectral Shape Analysis”. In: *CoRR* abs/1707.07070 (2017). arXiv: 1707.07070. url: http://arxiv.org/abs/1707.07070.

[48] Max Wardetzky et al. “Discrete Laplace operators: no free lunch”. In: *Symposium on Geometry processing*. Aire-la-Ville, Switzerland. 2007, pp. 33–37.

[49] Michael Zeltkevic. *Forward and Backward Euler Methods*. https://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node3.html.