Directional Graph Networks

Dominique Beaini [*1] **Saro Passaro** [*2] **Vincent Létourneau** [1 3] **William L. Hamilton** [3 4] **Gabriele Corso** [2]
Pietro Liò [2]

Abstract

The lack of anisotropic kernels in graph neural networks (GNNs) strongly limits their expressiveness, contributing to well-known issues such as over-smoothing. To overcome this limitation, we propose the first globally consistent anisotropic kernels for GNNs, allowing for graph convolutions that are defined according to topologicaly-derived directional flows. First, by defining a vector field in the graph, we develop a method of applying directional derivatives and smoothing by projecting node-specific messages into the field. Then, we propose the use of the Laplacian eigenvectors as such vector field. We show that the method generalizes CNNs on an n-dimensional grid and is provably more discriminative than standard GNNs regarding the Weisfeiler-Lehman 1-WL test. We evaluate our method on different standard benchmarks and see a relative error reduction of 8% on the CIFAR10 graph dataset and 11% to 32% on the molecular ZINC dataset, and a relative increase in precision of 1.6% on the MolPCBA dataset. An important outcome of this work is that it enables graph networks to embed directions in an unsupervised way, thus allowing a better representation of the anisotropic features in different physical or biological problems.

1. Introduction

One of the most important distinctions between convolutional neural networks (CNNs) and graph neural networks (GNNs) is that CNNs allow for any convolutional kernel, while most GNN methods are limited to symmetric kernels (also called isotropic kernels) (Kipf & Welling, 2016; Gilmer et al., 2017). There are some implementations of asymmetric kernels using gated mechanisms (Bresson & Laurent, 2017; Veličković et al., 2017), motif attention (Peng et al., 2019), edge features (Gilmer et al., 2017), port numbering (Sato et al., 2019) or the 3D structure of molecules (Klicpera et al., 2019).

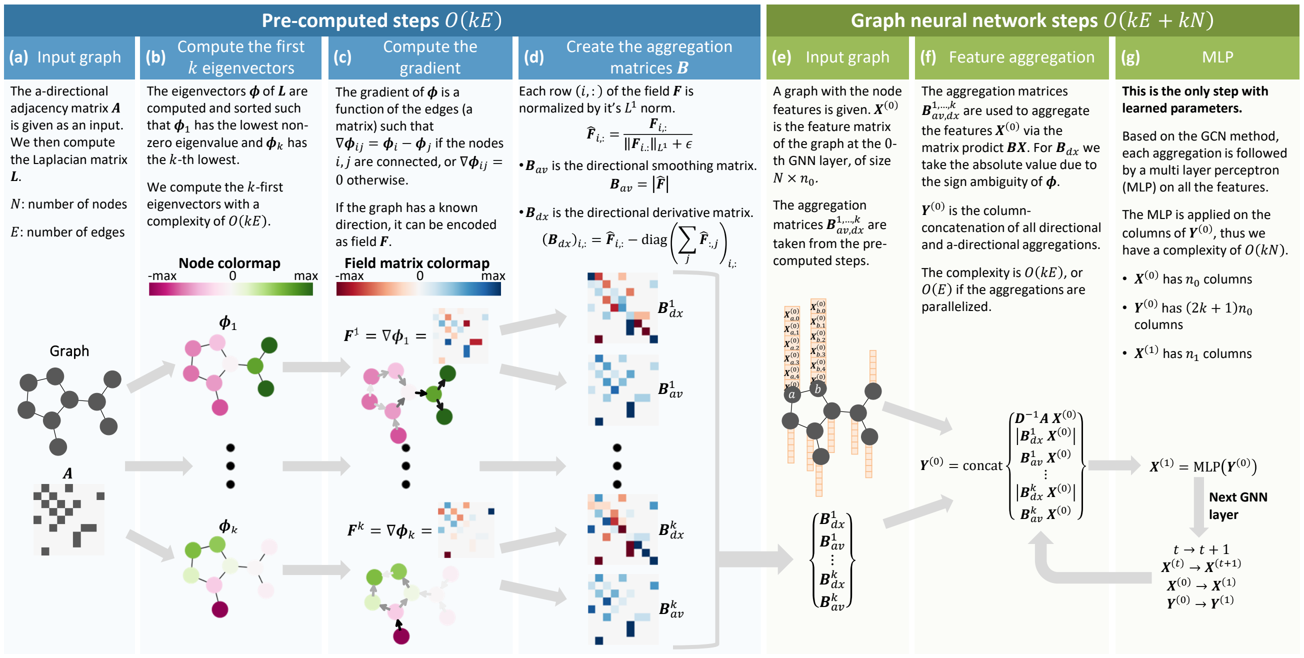
However, to the best of our knowledge, there are currently no methods that allow asymmetric graph kernels that are dependent on the full graph structure or directional flows. They either depend on local structures or local features. This is in opposition to images, which exhibit canonical directions: the horizontal and vertical axes. The absence of an analogous concept in graphs makes it difficult to define directional message passing and to produce an analogue of the directional frequency filters (or Gabor filters) widely present in image processing (Olah et al., 2020). In fact, there is numerous evidence that directional filtering is fundamental image processing (Kang et al., 2017; Antoine & Murenzi, 1996; Yue Lu & Minh N. Do, 2005).

We propose a novel idea for GNNs: use vector fields in the graph to define directions for the propagation of information. An overview of this framework is presented in figure 1. Using this approach, the usual message-passing structure of a GNN is projected onto globally-defined directions so that the contribution of each neighbouring node n_v is weighted by its alignment with the vector fields at the receiving node n_u. This enables our method to propagate information via directional derivatives or smoothing of the features.

In order to define globally consistent directional fields over general graphs, we propose to use the gradients of the low-frequency eigenvectors ϕ_k of the graph Laplacian, since they are known to capture key information about the global structure of graphs (Chavel, 1984; Chung et al., 1997; Grebenkov & Nguyen, 2013). In particular, these eigenvectors can be used to define optimal partitions of the nodes in a graph, to give a natural ordering (Levy, 2006), and to find the dominant directions of the graph diffusion process (Chung & Yau, 2000; Saerens et al., 2004). Further, we show that they generalize the horizontal and vertical directional flows in a grid (see figure 2), allowing them to guide the aggregation and mimic the asymmetric and directional kernels

[*]Equal contribution [1]InVivo AI, Montreal, Canada [2]department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom [3]MILA, Montreal, Canada [4]McGill University, Montreal, Canada. Correspondence to: Dominique Beaini <dominique@invivoai.com>, Saro Passaro <sp976@cam.ac.uk>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).


Pre-computed steps $O(kE)$
(a) Input graph
(b) Compute the first k eigenvectors
(c) Compute the gradient
(d) Create the aggregation matrices $\boldsymbol{B}$
Graph neural network steps $O(kE + kN)$
(e) Input graph
(f) Feature aggregation
(g) MLP
Node colormap
-max
0
max
Field matrix colormap
-max
0
max
Graph
$\boldsymbol{A}$
$\boldsymbol{\phi}_1$
$\boldsymbol{\phi}_k$
$\boldsymbol{F}^1 = \nabla\boldsymbol{\phi}_1 =$
$\boldsymbol{F}^k = \nabla\boldsymbol{\phi}_k =$
$\boldsymbol{B}^1_{dx}$
$\boldsymbol{B}^1_{av}$
$\boldsymbol{B}^k_{dx}$
$\boldsymbol{B}^k_{av}$
a
b
$\boldsymbol{X}^{(0)}_{a,0}$, $\boldsymbol{X}^{(0)}_{a,1}$, $\boldsymbol{X}^{(0)}_{a,2}$, $\boldsymbol{X}^{(0)}_{a,3}$, $\boldsymbol{X}^{(0)}_{a,4}$
$\boldsymbol{X}^{(0)}_{b,0}$, $\boldsymbol{X}^{(0)}_{b,1}$, $\boldsymbol{X}^{(0)}_{b,2}$, $\boldsymbol{X}^{(0)}_{b,3}$, $\boldsymbol{X}^{(0)}_{b,4}$
Next GNN layer

(a) The a-directional adjacency matrix $\boldsymbol{A}$ is given as an input. We then compute the Laplacian matrix $\boldsymbol{L}$.

N: number of nodes

E: number of edges

(b) The eigenvectors $\boldsymbol{\phi}$ of $\boldsymbol{L}$ are computed and sorted such that $\boldsymbol{\phi}_1$ has the lowest non-zero eigenvalue and $\boldsymbol{\phi}_k$ has the k-th lowest.

We compute the k-first eigenvectors with a complexity of $O(kE)$.

(c) The gradient of $\boldsymbol{\phi}$ is a function of the edges (a matrix) such that $\nabla\boldsymbol{\phi}_{ij} = \boldsymbol{\phi}_i - \boldsymbol{\phi}_j$ if the nodes i, j are connected, or $\nabla\boldsymbol{\phi}_{ij} = 0$ otherwise.

If the graph has a known direction, it can be encoded as field $\boldsymbol{F}$.

(d) Each row $(i,:)$ of the field $\boldsymbol{F}$ is normalized by it's L^1 norm.

$$\widehat{\boldsymbol{F}}_{i,:} = \frac{\boldsymbol{F}_{i,:}}{\|\boldsymbol{F}_{i,:}\|_{L^1} + \epsilon}$$

- $\boldsymbol{B}_{av}$ is the directional smoothing matrix.

$$\boldsymbol{B}_{av} = \left|\widehat{\boldsymbol{F}}\right|$$

- $\boldsymbol{B}_{dx}$ is the directional derivative matrix.

$$(\boldsymbol{B}_{dx})_{i,:} = \widehat{\boldsymbol{F}}_{i,:} - \mathrm{diag}\left(\sum_j \widehat{\boldsymbol{F}}_{:,j}\right)_{i,:}$$

(e) A graph with the node features is given. $\boldsymbol{X}^{(0)}$ is the feature matrix of the graph at the 0-th GNN layer, of size $N \times n_0$.

The aggregation matrices $\boldsymbol{B}^{1,\dots,k}_{av,dx}$ are taken from the pre-computed steps.

$$\begin{Bmatrix} \boldsymbol{B}^1_{dx} \\ \boldsymbol{B}^1_{av} \\ \vdots \\ \boldsymbol{B}^k_{dx} \\ \boldsymbol{B}^k_{av} \end{Bmatrix}$$

(f) The aggregation matrices $\boldsymbol{B}^{1,\dots,k}_{av,dx}$ are used to aggregate the features $\boldsymbol{X}^{(0)}$ via the matrix prodict $\boldsymbol{BX}$. For $\boldsymbol{B}_{dx}$ we take the absolute value due to the sign ambiguity of $\boldsymbol{\phi}$.

$\boldsymbol{Y}^{(0)}$ is the column-concatenation of all directional and a-directional aggregations.

The complexity is $O(kE)$, or $O(E)$ if the aggregations are parallelized.

$$\boldsymbol{Y}^{(0)} = \text{concat}\begin{Bmatrix} \boldsymbol{D}^{-1}\boldsymbol{A}\,\boldsymbol{X}^{(0)} \\ \left|\boldsymbol{B}^1_{dx}\,\boldsymbol{X}^{(0)}\right| \\ \boldsymbol{B}^1_{av}\,\boldsymbol{X}^{(0)} \\ \vdots \\ \left|\boldsymbol{B}^k_{dx}\,\boldsymbol{X}^{(0)}\right| \\ \boldsymbol{B}^k_{av}\,\boldsymbol{X}^{(0)} \end{Bmatrix}$$

(g) **This is the only step with learned parameters.**

Based on the GCN method, each aggregation is followed by a multi layer perceptron (MLP) on all the features.

The MLP is applied on the columns of $\boldsymbol{Y}^{(0)}$, thus we have a complexity of $O(kN)$.

- $\boldsymbol{X}^{(0)}$ has n_0 columns
- $\boldsymbol{Y}^{(0)}$ has $(2k+1)n_0$ columns
- $\boldsymbol{X}^{(1)}$ has n_1 columns

$$\boldsymbol{X}^{(1)} = \text{MLP}\left(\boldsymbol{Y}^{(0)}\right)$$

$t \to t+1$
$\boldsymbol{X}^{(t)} \to \boldsymbol{X}^{(t+1)}$
$\boldsymbol{X}^{(0)} \to \boldsymbol{X}^{(1)}$
$\boldsymbol{Y}^{(0)} \to \boldsymbol{Y}^{(1)}$

Figure 1. Overview of the steps required to aggregate messages in the direction of the eigenvectors.

present in computer vision. In fact, we demonstrate mathematically that our work generalizes CNNs, by reproducing all convolutional kernels of radius R in an n-dimensional grid, while also bringing the powerful data augmentation capabilities of reflection, rotation or distortion of the directions. Additionally, we also prove that our directional graph networks (DGNs) are more discriminative than standard GNNs in regards to the Weisfeiler-Lehman 1-WL test, confirming an increase of expressiveness.

We further show that our DGN model theoretically and empirically allows for efficient message passing across distant communities, which counteracts the well-known problem of over-smoothing in GNNs. Alternative methods reduce the impact of over-smoothing by using skip connections (Luan et al., 2019), global pooling (Alon & Yahav, 2020), or randomly dropping edges during training time (Rong et al., 2020), but without solving the underlying problem.

Our method distinguishes itself from other spectral GNNs since the literature usually uses the low frequencies to estimate local Fourier transforms in the graph (Levie et al., 2018; Xu et al., 2019). Instead, we do not try to approximate the Fourier transform, but only to define a directional flow at each node and guide the aggregation.

We tested our method on 5 standard datasets from (Dwivedi et al., 2020) and (Hu et al., 2020), using two types of architectures, and either using or ignoring edge features. In all cases, we observed state-of-the-art results from the proposed DGN, with relative improvements of 8% on CIFAR10, 11-32% on ZINC, 0.8% on MolHIV and 1.6% on MolPCBA. Most of the improvement is attributed to the directional derivative aggregator, highlighting our method's ability of capturing directional high-frequency signals in graphs.

2. Theoretical development

2.1. Intuitive overview

One of the biggest limitations of current GNN methods compared to CNNs is the inability to do message passing in a specific direction such as the horizontal one in a grid graph. In fact, it is difficult to define directions or coordinates based solely on the shape of the graph.

The lack of directions strongly limits the discriminative abilities of GNNs to understand local structures and simple feature transformations. Most GNNs are invariant to the permutation of the neighbours' features, so the nodes' received signal is not influenced by swapping the features of two neighbours. Therefore, several layers in a deep network will be employed to understand these simple changes instead of being used for higher level features, leading to problematic phenomena such as a over-squashing (Alon & Yahav, 2020).

In the first part of the theoretical development, we develop the mathematical theory for general vector fields $\boldsymbol{F}$. Intuitively, defining a vector field over a graph corresponds to assigning a scalar weight to edges corresponding to the magnitude of the flow in that direction. Note that $\boldsymbol{F}$ has the same shape as the adjacency matrix and the same zero entries. As an example a left-to-right flow in a grid corresponds to a matrix with positive values over all left-to-right

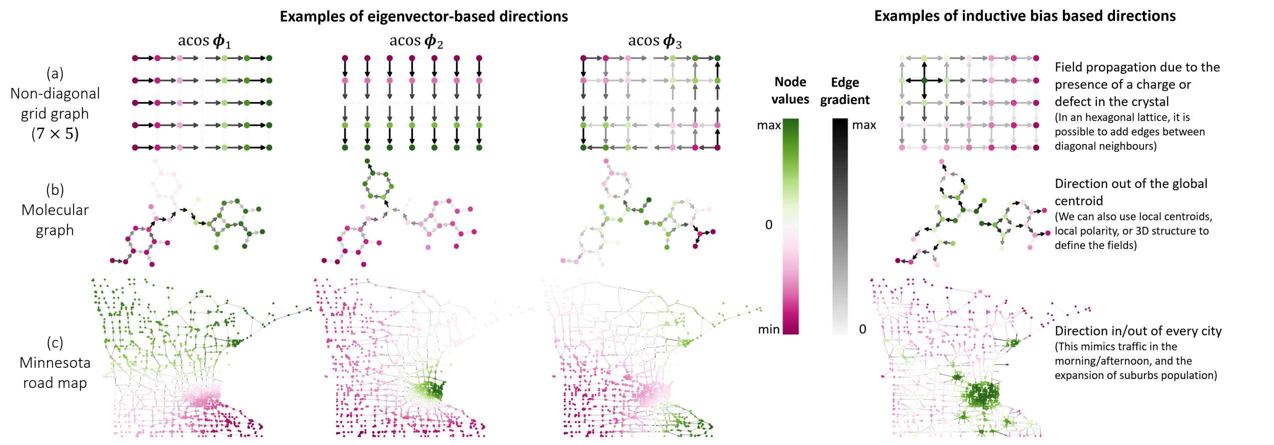

Examples of eigenvector-based directions
acos ϕ_1
acos ϕ_2
acos ϕ_3
Examples of inductive bias based directions
(a) Non-diagonal grid graph (7 × 5)
(b) Molecular graph
(c) Minnesota road map
Node values
Edge gradient
max
0
min
max
0
Field propagation due to the presence of a charge or defect in the crystal
(In an hexagonal lattice, it is possible to add edges between diagonal neighbours)
Direction out of the global centroid
(We can also use local centroids, local polarity, or 3D structure to define the fields)
Direction in/out of every city
(This mimics traffic in the morning/afternoon, and the expansion of suburbs population)

Figure 2. Possible directional flows in different types of graphs. The node coloring is a potential map and the edges represent the gradient of the potential with the arrows in the direction of the flow. The first 3 columns present the arcosine of the normalized eigenvectors (acos $\hat{\phi}$) as node coloring, and their gradients represented as edge intensity. The last column presents examples of inductive bias introduced in the choice of direction. (a) The eigenvectors 1 and 2 are the horizontal and vertical flows of the grid. (b) The eigenvectors 1 and 2 are the flow in the longest and second-longest directions. (c) The eigenvectors 1, 2 and 3 flow respectively in the South-North, suburbs to the city center and West-East directions. We ignore ϕ_0 since it is constant and has no direction.

edges, negative over the right-to-left edges and 0 on the vertical edges.

In the second part, we set $\boldsymbol{F}$ to be the gradient of the low-frequency eigenvectors of the Laplacian. Using this directional field, we show that the expressiveness of GNNs can be improved, while providing an intuitive directional flows over a variety of graphs (see figure 2). For example, we prove that in grid-shaped graphs some of these eigenvectors correspond to the horizontal and vertical flows. Again, we observe in the Minnesota map that the first 3 non-constant eigenvectors produce logical directions, namely South/North, suburb/city, and West/East.

Another important contribution—also noted in figure 2—is the ability to define any kind of directional flow based on prior knowledge of the problem. Hence, instead of relying on eigenvectors to find directions in a map, we can simply use the cardinal directions or the rush-hour traffic flow.

2.2. Overview of the theoretical contributions

Vector fields in a graph. Using directions in a graph is novel and not intuitive, so our first step is to define a simple nomenclature where we use a vector field to define a directional flow at each node.

Directional smoothing and derivatives. To make use of vector fields over graphs, we define aggregation matrices that can either smooth the signal (low pass filter) or compute its derivative (high pass filter) according to the directions specified by the vector field.

Gradient of the Laplacian eigenvectors. We show that using the gradient of the low-frequency eigenvectors of the graph Laplacian generates interpretable vector fields that counteract the over-smoothing problem.

Generalization of CNNs. We demonstrate that, when applied to a grid graph, the eigenvector-based directional aggregation generalizes convolutional neural networks.

Comparison to the Weisfeiler-Lehman (WL) test. We prove that the proposed DGN is more expressive than the 1-WL test, and thus more expressive than ordinary GNNs.

2.3. Vector fields in a graph

This section presents the ideas of differential geometry applied to graphs, with the goal of finding proper definitions of scalar products, gradients and directional derivatives. For reference see for example (Bronstein et al., 2017; Grebenkov & Nguyen, 2013; Grady & Polimeni, 2010).

Let $G = (V, E)$ be a graph with V the set of vertices and $E \subset V \times V$ the set of edges. The graph is undirected meaning that $(i, j) \in E$ iff $(j, i) \in E$. Define the vector spaces $L^2(V)$ and $L^2(E)$ as the set of maps $V \to \mathbb{R}$ and $E \to \mathbb{R}$ with $\boldsymbol{x}, \boldsymbol{y} \in L^2(V)$ and $\boldsymbol{F}, \boldsymbol{H} \in L^2(E)$ and scalar products

$$\begin{aligned} \langle \boldsymbol{x}, \boldsymbol{y} \rangle_{L^2(V)} &:= \sum_{i \in V} \boldsymbol{x}_i \boldsymbol{y}_i \\ \langle \boldsymbol{F}, \boldsymbol{H} \rangle_{L^2(E)} &:= \sum_{(i,j) \in E} \boldsymbol{F}_{(i,j)} \boldsymbol{H}_{(i,j)} \end{aligned} \tag{1}$$

Think of E as the "tangent space" to V and of $L^2(E)$ as the set of "vector fields" on the space V with each row $\boldsymbol{F}_{i,:}$ representing a vector at the i-th node, and the element $\boldsymbol{F}_{i,j}$

being the component of the vector going from node i to j through edge e_{ij}. Note that with n the number of nodes in G, any $\boldsymbol{x} \in L^2(V)$ can be represented as an n coordinates vector and $\boldsymbol{F} \in L^2(E)$ can be represented as an $n \times n$ matrix.

Define the pointwise scalar product as the map $L^2(E) \times L^2(E) \to L^2(V)$ taking 2 vector fields and returning their inner product at each point of V, at the node i is defined by equation 2.

$$\langle \boldsymbol{F}, \boldsymbol{H} \rangle_i := \sum_{j:(i,j)\in E} \boldsymbol{F}_{i,j} \boldsymbol{H}_{i,j} \tag{2}$$

In equation 3, we define the gradient ∇ as a mapping $L^2(V) \to L^2(E)$ and the divergence div as a mapping $L^2(E) \to L^2(V)$, thus leading to an analogue of the directional derivative in equation 4.

$$\begin{aligned} (\nabla \boldsymbol{x})_{(i,j)} &:= \boldsymbol{x}(j) - \boldsymbol{x}(i) \\ (\operatorname{div} \boldsymbol{F})_i &:= \sum_{j:(i,j)\in E} \boldsymbol{F}_{(i,j)} \end{aligned} \tag{3}$$

Definition 1. *The directional derivative of the function $\boldsymbol{x}$ on the graph G in the direction of the vector field $\hat{\boldsymbol{F}}$ where each vector is of unit-norm is*

$$D_{\hat{\boldsymbol{F}}} \boldsymbol{x}(i) := \langle \nabla \boldsymbol{x}, \hat{\boldsymbol{F}} \rangle_i = \sum_{j:(i,j)\in E} (\boldsymbol{x}(j) - \boldsymbol{x}(i)) \hat{\boldsymbol{F}}_{i,j} \tag{4}$$

$|\boldsymbol{F}|$ will denote the absolute value of $\boldsymbol{F}$ and $||\boldsymbol{F}_{i,:}||_{L^p}$ the L^p-norm of the i-th row of $\boldsymbol{F}$. We also define the forward/backward directions as the positive/negative parts of the field $\boldsymbol{F}^{\pm}$.

2.4. Directional smoothing and derivatives

Next, we show how the vector field $\boldsymbol{F}$ is used to *guide* the graph aggregation by projecting the incoming messages. Specifically, we define the weighted aggregation matrices $\boldsymbol{B}_{av}$ and $\boldsymbol{B}_{dx}$ that allow to compute the directional smoothing and directional derivative of the node features, as presented visually in figure 1-d.

The directional average matrix $\boldsymbol{B}_{av}$ is the weighted aggregation matrix such that all weights are positives and all rows have an L^1-norm equal to 1, as shown in equation 5 and theorem 2.1, with a proof in the appendix D.1.

$$\boldsymbol{B}_{av}(\boldsymbol{F})_{i,:} = \frac{|\boldsymbol{F}_{i,:}|}{||\boldsymbol{F}_{i,:}||_{L^1} + \epsilon} \tag{5}$$

The variable ϵ is an arbitrarily small positive number used to avoid floating-point errors. The L^1-norm denominator is a local row-wise normalization. The aggregator works by assigning a large weight to the elements in the forward or backward direction of the field, while assigning a small weight to the other elements, with a total weight of 1.

Theorem 2.1 (Directional smoothing)**.** *The operation $\boldsymbol{y} = \boldsymbol{B}_{av}\boldsymbol{x}$ is the directional average of $\boldsymbol{x}$, in the sense that $\boldsymbol{y}_u$ is the mean of $\boldsymbol{x}_v$, weighted by the direction and amplitude of $\boldsymbol{F}$.*

With $\boldsymbol{x}_v$ the features at the nodes v neighbouring u, and $\boldsymbol{y}_u$ the directional smoothing at node u.

The directional derivative matrix $\boldsymbol{B}_{dx}$ is defined in (6) and theorem 2.2, with the proof in appendix D.2. Again, the denominator is a local row-wise normalization but can be replaced by a global normalization. diag$(\boldsymbol{a})$ is a square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$. The aggregator works by subtracting the projected forward message by the backward message (similar to a center derivative), with an additional diagonal term to balance both directions.

$$\begin{aligned} \boldsymbol{B}_{dx}(\boldsymbol{F})_{i,:} &= \hat{\boldsymbol{F}}_{i,:} - \operatorname{diag}\Big(\sum_j \hat{\boldsymbol{F}}_{:,j}\Big)_{i,:} \\ \hat{\boldsymbol{F}}_{i,:} &= \left(\frac{\boldsymbol{F}_{i,:}}{||\boldsymbol{F}_{i,:}||_{L^1} + \epsilon}\right) \end{aligned} \tag{6}$$

Theorem 2.2 (Directional derivative)**.** *Suppose $\hat{\boldsymbol{F}}$ have rows of unit L^1 norm. The operation $\boldsymbol{y} = \boldsymbol{B}_{dx}(\hat{\boldsymbol{F}})\boldsymbol{x}$ is the centered directional derivative of $\boldsymbol{x}$ in the direction of $\boldsymbol{F}$, in the sense of equation 4, i.e.*

$$\boldsymbol{y} = D_{\hat{\boldsymbol{F}}} \boldsymbol{x} = \Big(\hat{\boldsymbol{F}} - \operatorname{diag}\Big(\sum_j \hat{\boldsymbol{F}}_{:,j}\Big)\Big)\boldsymbol{x}$$

These aggregators are directional, interpretable and complementary, making them ideal choices for GNNs. We discuss the choice of aggregators in more details in appendix A, while also providing alternative aggregation matrices such as the center-balanced smoothing, the forward-copy, the phantom zero-padding, and the hardening of the aggregators using softmax/argmax on the field. We further provide a visual interpretation of the $\boldsymbol{B}_{av}$ and $\boldsymbol{B}_{dx}$ aggregators in figure 3. Interestingly, we also note in appendix A.1 that $\boldsymbol{B}_{av}$ and $\boldsymbol{B}_{dx}$ yield respectively the mean and Laplacian aggregations when $\boldsymbol{F}$ is a vector field such that all entries are constant $\boldsymbol{F}_{ij} = \pm C$.

2.5. Gradient of the Laplacian eigenvectors as interpretable vector fields

In this section we give theoretical support for the choice of gradients of the eigenfunctions of the Laplacian as sensible vectors along which to do directional message passing since they are interpretable and allow to reduce the over-smoothing. This section gives a theoretical ground to the

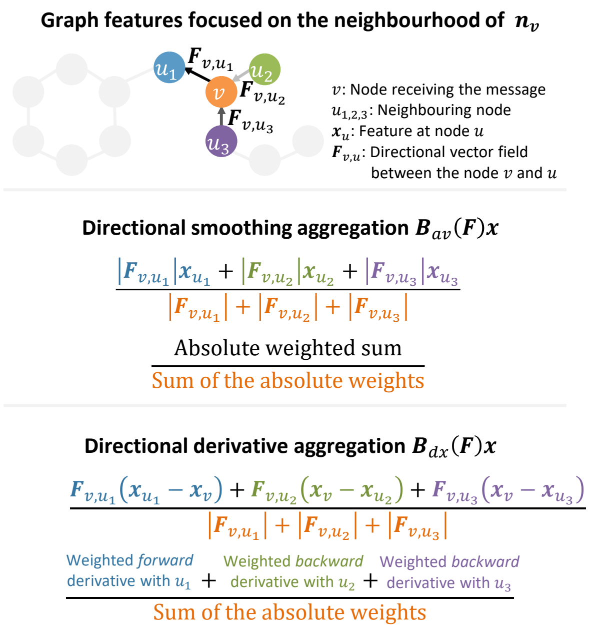

Graph features focused on the neighbourhood of n_v
u_1
F_{v,u_1}
u_2
v
F_{v,u_2}
F_{v,u_3}
u_3
v: Node receiving the message
$u_{1,2,3}$: Neighbouring node
x_u: Feature at node u
$F_{v,u}$: Directional vector field between the node v and u
Directional smoothing aggregation $B_{av}(F)x$
$\frac{|F_{v,u_1}|x_{u_1} + |F_{v,u_2}|x_{u_2} + |F_{v,u_3}|x_{u_3}}{|F_{v,u_1}| + |F_{v,u_2}| + |F_{v,u_3}|}$
Absolute weighted sum
Sum of the absolute weights
Directional derivative aggregation $B_{dx}(F)x$
$\frac{F_{v,u_1}(x_{u_1} - x_v) + F_{v,u_2}(x_v - x_{u_2}) + F_{v,u_3}(x_v - x_{u_3})}{|F_{v,u_1}| + |F_{v,u_2}| + |F_{v,u_3}|}$
Weighted forward derivative with u_1 +
Weighted backward derivative with u_2 +
Weighted backward derivative with u_3
Sum of the absolute weights

Figure 3. Illustration of how the directional aggregation works at a node n_v, with the arrows representing the direction and intensity of the field $\boldsymbol{F}$.

intuitive directions presented in figure 2, and is the motivation behind steps (b-c) in figure 1.

As usual the combinatorial, degree-normalized and symmetric normalized Laplacian are defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}, \quad \boldsymbol{L}_{\text{norm}} = \boldsymbol{D}^{-1}\boldsymbol{L}, \quad \boldsymbol{L}_{\text{sym}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}} \tag{7}$$

The eigenvectors of these matrices are known to capture many essential properties of graphs, making them a natural foundation for directional message passing. For example, the Laplacian eigenvectors corresponding to the smallest eigenvalues (i.e., the low frequency eigenvectors) effectively capture the community structure of a graph, and these eigenvectors also play the role of Fourier modes in graph signal processing (Hamilton, 2020). Indeed, the Laplacian eigenvectors hold such rich information about graph structure that their study is the focus of the mathematical subfield of spectral graph theory (Chung et al., 1997).

In order to illustrate the utility of these eigenvectors in the context of GNNs, we show that the low-frequency eigenvectors provide a natural direction that allows us to pass messages between distant nodes in a graph. In particular, we show in theorem 2.3 (proved in appendix D.3) that by passing information in the direction of $\boldsymbol{\phi}_1$, the eigenvector associated to the lowest non-trivial frequency of $\boldsymbol{L}_{\text{norm}}$, DGNs can efficiently share information between distant nodes of the graph by reducing the diffusion distance between them. This idea is reflected in figure 2, where we see that the eigenvectors of the Laplacian give directions that correspond to a natural notion of distance on real-world graphs.

In the next paragraphs, we will prove that following the gradient of the eigenvectors allows to effectively reduce the heat-kernel distance between pairs of nodes.

Consider the transition matrix $\boldsymbol{W} = \boldsymbol{D}^{-1}\boldsymbol{A}$. Its entries can be used to define a random walk with probability to move from node x to node y equal to $p_1(x,y) = \frac{1}{d_x}$ if x and y are neighbors and 0 if not. Notice that the probability to transition from x to y in k steps is given by the x, y entry of the matrix $\boldsymbol{W}^k$. This matrix is also called the discrete heat kernel $p_k(x,y) = (\boldsymbol{W}^k)_{x,y}$. Given a Markov process $\tilde{X}_k$ defined by the transition matrices $\boldsymbol{W}^k$, $j = 1, ..., k$, we can define a continuous time random walk on the same graph in the following way. Let N_t be a mean 1 Poisson random variable, the continuous time random variable is defined by $X_t := \tilde{X}_{N_t}$ with transition probability $q_t(x,y) = P(X_t = y|x_0 = x)$.

In (Barlow, 2017), the following identity is shown

$$q_t(x,y) = \sum_{n=0}^{\infty} \frac{e^{-t}t^k}{k!} p_k(x,y)$$

Or in matrix form $q_t = e^{t(\boldsymbol{W}-\boldsymbol{I})} = e^{-t\boldsymbol{L}_{\text{norm}}}$. This transition probability is also called the continuous time heat kernel because it satisfies the continuous time heat equation on graphs $\frac{d}{dt}q_t = -\boldsymbol{L}_{\text{norm}}q_t$. In (Coifman & Lafon, 2006) the following distance is defined

Definition 2 (Diffusion distance)**.** *The diffusion distance at time t between the nodes x, y is*

$$d_t(x,y) := \left(\sum_{z \in V} \Big(q_t(x,z) - q_t(y,z)\Big)^2\right)^{\frac{1}{2}} \tag{8}$$

The diffusion distance is small when there is high probability that two random walks starting at x and y meet at time t. The diffusion distance is used as a model of how the data at a node x influences a node y in a GNN. The symmetrisation of the heat kernel in the diffusion distance and the use of continuous time are slight departure from the actual process of information diffusion in a GNN but allow us to describe the important phenomenons with much simpler statements.

Definition 3 (Gradient step)**.** *Suppose the two neighboring nodes x and z are such that $\phi(z) - \phi(x)$ is maximal among the neighbors of x, then we will say z is obtained from x by taking a step in the direction of the gradient $\nabla\phi$.*

Theorem 2.3 (Gradient steps reduce diffusion distance)**.** *Let x, y be nodes such that $\phi_1(x) < \phi_1(y)$. Let x' be the node obtained from x by taking one step in the direction of $\nabla\phi_1$, then there is a constant C such that for $C \leq t$ we have*

$$d_t(x', y) < d_t(x,y).$$

With the reduction in distance being proportional to $e^{-\lambda_1}$.

From this theorem, we see that moving from node x to node x' by following the gradient of the eigenvector $\boldsymbol{\phi}_1$ is guaranteed to reduce the heat kernel distance with a destination node y. While the theorem always holds for $\boldsymbol{\phi}_1$, it should be true for higher frequency eigenvectors if the graph has added structure for example if it is an approximation of a surface or a higher dimensional manifold.

In the context of GNNs, Theorem 2.3 also has implications for the well-known problems of *over-smoothing* and *over-squashing* (Alon & Yahav, 2020; Hamilton, 2020). In most GNN models, node representations become over-smoothed after several rounds of message passing, as the representations tend to reach a mean-field equilibrium equivalent to the stationary distribution of a random walk (Hamilton, 2020). Researchers have also highlighted the related issue of over-squashing, which reflects the inability for GNNs to propagate informative signals between distant nodes in a graph (Alon & Yahav, 2020).

Both these problems are related to the fact that the influence of one node's input on the final representation of another node in a GNN is correlated with the diffusion distance between the nodes (Xu et al., 2018b). Theorem 2.3 highlights how the DGN approach can alleviate these issues. In particular, the Laplacian eigenfunctions reveal directions that can counteract over-smoothing and over-squashing by allowing efficient propagation of information between distant nodes instead of following a diffusion process.

Finally it is interesting to note that by selecting different eigenvectors as basis of directions, our method further aligns with a theorem that multiple independent aggregators are needed to distinguish neighbourhoods of nodes with continuous features (Corso et al., 2020).

2.6. Choosing a basis of the Laplacian eigenspace

When using eigenvectors of the Laplacian $\boldsymbol{\phi}_i$ to define directions in a graph, we need to keep in mind that there is never a single eigenvector associated to an eigenvalue, but a whole eigenspace. If an eigenvalue has multiplicity of k, the associated eigenspace has dimension k and any collection of k orthogonal vectors could be chosen as basis of that space and as vectors for the definitions of the aggregation matrices $\boldsymbol{B}$ defined in the previous sections.

Disconnected graphs. When a graph is disconnected, then the eigenfunctions will simply be the combination of the eigenfunctions of each connected components. Hence, one must consider $\boldsymbol{\phi}_i$ as the i-th eigenvector of each component when taken separately.

Normalizing the eigenvectors. For an eigenvalue of multiplicity 1, there are always two unit norm eigenvectors of opposite sign, which poses a problem during the directional aggregation. We can make a choice of sign and later take the absolute value (i.e. $\boldsymbol{B}_{av}$ in equation 5). An alternative that applies to multiplicities higher than 1 is to take samples of orthonormal bases of the eigenspace and use each choice to augment the training (see section 2.10).

Multiplicities greater than 1. Although multiplicities higher than one do happen for low-frequencies (square grids have a multiplicity 2 for λ_1) this is not common in "real-world graphs" since it suggests symmetries in the graph which are uncommon. Furthermore, we found no λ_1 multiplicity greater than 1 in the ZINC and PATTERN datasets. We further discuss these rare cases and how to deal with them in appendix C.4.

Orthogonal directions. Although all $\boldsymbol{\phi}$ are orthogonal, their gradients, used to define directions, are not always *locally* orthogonal (e.g. there are many horizontal flows in the grid). This concern is left to be addressed in future work.

2.7. Generalization of the convolution on a grid

In this section we show that our method generalizes CNNs by allowing to define any radius-R convolutional kernels in grid-shaped graphs. The radius-R kernel at node u is a convolutional kernel that takes the weighted sum of all nodes v at a distance $d(u, v) \leq R$.

Consider the lattice graph Γ of size $N_1 \times N_2 \times ... \times N_n$ where each vertices are connected to their direct non-diagonal neighbour. We know from Lemma D.1 that, for each dimension, there is an eigenvector that is only a function of this specific dimension. For example, the lowest frequency eigenvector $\boldsymbol{\phi}_1$ always flows in the direction of the longest length. Hence, the Laplacian eigenvectors of the grid can play a role analogous to the axes in Euclidean space, as shown in figure 2.

With this knowledge, we show in theorem 2.4 (proven in D.6), that we can generalize all convolutional kernels in an n-dimensional grid. This is a strong result since it demonstrates that our DGN framework generalizes CNNs when applied on a grid, thus closing the gap between GNNs and the highly successful CNNs on image tasks.

Theorem 2.4 (Generalization radius-R convolutional kernel in a lattice)**.** *For an* n*-dimensional lattice, any convolutional kernel of radius* R *can be realized by a linear combination of directional aggregation matrices and their compositions.*

As an example, figure 4 shows how a linear combination of the first and m-th aggregators $\boldsymbol{B}(\nabla\boldsymbol{\phi}_{1,m})$ realize a kernel on an $N \times M$ grid, where $m = \lceil N/M \rceil$ and $N > M$.

Note that when the size of a given dimension is an integer multiple of another direction, e.g. $N = M$ or $N = 3M$, then you will find a multiplicity of 2 for the $m - th$ eigenvector. Hence, the eigenvector used to define the direction is not unique. This does not void theorem 2.4 since the eigenvectors flowing in the horizontal/vertical directions are still valid choices.

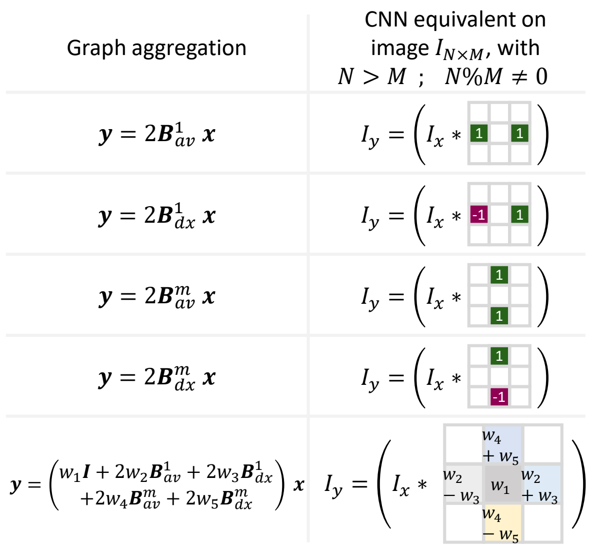


Figure 4. Realization of a radius-1 convolution using the proposed aggregators. I_x is the input feature map, $*$ the convolutional operator, I_y the convolution result, and $\boldsymbol{B}^i = \boldsymbol{B}(\nabla\phi_i)$.

2.8. Extending the radius of the aggregation kernel

Having aggregation kernels for neighbours of distance 2 or 3 is important to improve the expressiveness of GNNs, their ability to understand patterns, and to reduce the number of layers required. However, the lack of directions in GNNs strongly limits the radius of the kernels since, given a graph of regular degree d, a mean/sum aggregation at a radius-R will result in a heavy over-squashing of $O(d^R)$ messages. Using the directional fields, we can enumerate different paths, thus assigning a different weight for different R-distant neighbours. This method, proposed in appendix A.7, avoids the over-squashing. (Empirical results on this extension are left for future work.)

2.9. Comparison with Weisfeiler-Lehman (WL) test

We also compare the expressiveness of the Directional Graph Networks with the classical WL graph isomorphism test which is often used to classify the expressivity of graph neural networks (Xu et al., 2018a). In theorem 2.5 (proven in appendix D.7) we show that DGNs are capable of distinguishing pairs of graphs that the 1-WL test (and so ordinary GNNs) cannot differentiate.

Theorem 2.5 (Comparison with 1-WL test)**.** *DGNs using the mean aggregator, any directional aggregator of the first Laplacian eigenvector and injective degree-scalers are strictly more powerful than the 1-WL test.*

2.10. Data augmentation

Another theoretical result is that the directions in the graph allow to replicate some of the most common data augmentation techniques used in computer vision, namely reflection, rotation and distortion. The main difference is that, instead of modifying the image (such as a $5°$ rotation), the proposed transformation is applied on the vector field defining the aggregation kernel (thus rotating the kernel by $-5°$ without changing the image). This offers the advantage of avoiding to pre-process the data since the augmentation is done directly on the kernel at each iteration of the training.

In appendix B.1, we explain how we can generalize some image augmentation techniques by defining the reflection, rotation and distortion augmentations in our DGN framework, and provide early results on the CIFAR10 dataset in appendix B.2.

3. Implementation

We implemented the models using the DGL and PyTorch libraries and we provide the code at the address https://github.com/Saro00/DGN. We test our method on standard benchmarks from (Dwivedi et al., 2020) and (Hu et al., 2020), namely ZINC, CIFAR10, PATTERN, MolHIV and MolPCBA with more details on the datasets and how we enforce a fair comparison in appendix C.1.

For the empirical experiments we inserted our proposed aggregation method in two different type of message passing architectures used in the literature: a *simple* convolutional architecture similar to the one present in GCN (equation 9a) (Kipf & Welling, 2016) and a more *complex* and general one typical of MPNNs (9b) (Gilmer et al., 2017) with or without edge features e_{ji}. The time complexity of our approach is $O(Em)$, which is identical to PNA (Corso et al., 2020), where E is the number of edges and m the number of aggregators, with an additional $O(Ek)$ to pre-compute the k-first eigenvectors, as explained in the appendix C.2.

$$X_i^{(t+1)} = U\left(\bigoplus_{(j,i)\in E} X_j^{(t)}\right) \tag{9a}$$

$$X_i^{(t+1)} = U\left(X_i^{(t)}, \bigoplus_{(j,i)\in E} M\left(X_i^{(t)}, X_j^{(t)}, \underbrace{e_{ji}}_{\text{optional}}\right)\right) \tag{9b}$$

Here, $\bigoplus$ is an operator which concatenates the results of multiple aggregators, X is the node features, M is a linear transformation and U a multiple layer perceptron (MLP). This *simple* architecture of equation 9a is observed visually in steps (f-g) of figure 1.

We further use degree scalers $S(d, \alpha)$ defined below to scale the aggregation results according to each node's degree, as proposed by the PNA model (Corso et al., 2020). Here, d is the degree of a given node, δ is the average node degree in the training set, and α is a parameter set to -1 for degree-attenuation and 1 for degree amplification. Note that each degree scaler is applied to the result of each aggregator, and the results are concatenated.

$$S(d, \alpha) = \left(\frac{\log(d+1)}{\delta}\right)^{\alpha}, \ \delta = \frac{1}{|\text{train}|} \sum_{i \,\in\, \text{train}} \log(d_i + 1) \tag{10}$$

We tested the directional aggregators across the datasets using the gradient of the first k eigenvectors $\nabla \boldsymbol{\phi}_{1,\ldots,k}$ as the underlying vector fields. Here, k is a hyperparameter, usually 1 or 2, but could be bigger for high-dimensional graphs. To deal with the arbitrary sign of the eigenvectors, we take the absolute value of the result of equation 6, making it invariant to a reflection of the field. In case of a disconnected graph, $\boldsymbol{\phi}_i$ is the i-th eigenvector of each connected component. Despite the numerous aggregators proposed in appendix A, only $\boldsymbol{B}_{dx}$ and $\boldsymbol{B}_{av}$ are tested empirically.

The metrics used to measure the performance of a model depend are enforced for each dataset and provided by (Dwivedi et al., 2020) and (Hu et al., 2020). In particular, we use the mean absolute error (MAE), the accuracy (acc), the area under the receiver operating curve (ROC-AUC), and the average precision (AP).

4. Results and discussion

Directional aggregation Using the benchmarks introduced in section 3, we present in figure 5 a fair comparison of various aggregation strategies using the same parameter budget and hyperparameters. We see a consistent boost in the performance for *simple*, *complex* and *complex with edges* models using directional aggregators compared to the *mean-aggregator* baseline.

With our theoretical analysis in mind, we expected to perform well on PATTERN since the flow of the first eigenvectors are meaningful directions in a stochastic block model (i.e., these eigenvectors tend to correlate with community membership). The results match our expectations, outperforming all the previous models.

In particular, we see a significant improvement in the molecular datasets (ZINC, MolHIV and MolPCBA) when using the directional aggregators, especially for the derivative aggregation $\boldsymbol{B}^1_{dx}$ (noted *dx1* in figure 5). We believe this is due to the capacity to efficiently move messages across opposite parts of the molecule and to better understand the role of atom pairs. We further believe that the derivative aggregator is better able to capture high-frequency directional signals, similarly to the Gabor filters in computer vision.

Further, the thesis that DGNs can bridge the gap between CNNs and GNNs is supported by the clear improvements on CIFAR10 over the baselines.

Improvements over positional embeddings. In the work by (Dwivedi et al., 2020), they proposed the use of positional encoding of the eigenvectors. However, our experiments with the positional encoding of the first 2 non-trivial eigenvectors, noted *pos1*, *pos2* in figure 5, showed no clear improvement on most datasets. In fact, Dwivedi et al. noted that many eigenvectors and high network depths are required for improvements, yet we outperform their results with fewer parameters, less depth, and only 1-2 eigenvectors, further motivating their use as directional flows instead of positional encoding.

Comparison to the literature. In order to compare our model with the literature, we fine-tuned it on the various datasets and we report its performance in figure 6. We observe that DGN provides significant improvement across all benchmarks, highlighting the importance of anisotropic kernels that are dependant on the graph topology.

Note that the results in Figure 6 are better those in Figure 5 since the latter uses a more exhaustive parameter search, and uses the *min/max* aggregators proposed in PNA (Corso et al., 2020) alongside the directional aggregators.

5. Conclusion

The proposed DGN method allows to address many problems of GNNs, including the lack of anisotropy, the low expressiveness, the over-smoothing and over-squashing. For the first time in graph networks, we generalize the directional properties of CNNs and their data augmentation capabilities. Based on the intuitive idea that the low-frequency eigenvectors of the graph Laplacian gives an interpretable directional flow, we backed our work by a set of strong theoretical results showing that these eigenvectors are important in connecting nodes that are far away and improving the expressiveness in regards to the WL-test.

The work being also supported by strong empirical results, we believe it will give rise to a new family of directional GNNs. In fact, we introduce in the appendix different avenues for future work, including the hardening of the aggregators A.4, the introduction of a zero-padding at the boundaries A.6, the implementation of radius-R kernels A.7, and the full study of directional data augmentation B.1. Future methods could also improve the choice of multiple directions beyond the selection of the *k-lowest* frequencies.

Broader Impact. This work will extend the usability of graph networks to all problems with engineering and physically defined directions, thus making GNN a new laboratory for signal processing, physics, material science and molecular and cell biology. In fact, the anisotropy present

	ZINC			PATTERN		CIFAR10		MolHIV	MolPCBA	
Aggregators	Simple MAE	Complex MAE	Complex-E MAE	Simple % acc	Complex % acc	Simple % acc	Complex % acc	Simple % ROC-AUC	Complex % AP	Complex-E % AP
mean	0.316	0.353	0.262	80.77	83.34	55.9	62.8	75.1	26.04	26.38
mean pos_1	0.349	0.332	0.297	80.76	83.74			75.8	26.97	27.50
mean pos_1 pos_2	0.344	0.330	0.284	84.51	81.25			76.1	26.03	25.65
mean dx_1	0.296	**0.233**	**0.191**	84.22	83.44			78.0	26.79	**27.91**
mean dx_1 dx_2	0.337	0.271	0.205	81.61	86.62	52.9	**69.8**	76.5	**27.16**	26.55
mean av_1	0.317	0.332	0.276	84.54	83.21			78.4	25.97	26.66
mean av_1 av_2	0.367	0.332	0.260	85.12	85.38	**60.6**	65.1	77.1	25.61	26.67
mean dx_1 av_1	**0.290**	0.245	0.192	**85.17**	**86.68**			**79.0**	26.40	27.47

Best

Worst

Figure 5. Test set results using a parameter budget of $\sim 100k$ with the same hyperparameters as (Corso et al., 2020), except MolPCBA with a budget of $\sim 7M$. The low-frequency Laplacian eigenvectors are used to define the directions, except for CIFAR10 that uses the coordinates of the image. For brevity, we denote dx_i and av_i as the directional derivative $\boldsymbol{B}^i_{dx}$ and smoothing $\boldsymbol{B}^i_{av}$ aggregators of the i-th direction. We also denote pos_i as the i-th eigenvector used as positional encoding for the mean aggregator.

	ZINC		PATTERN	CIFAR10		MolHIV	MolPCBA
Model	No edge features MAE	Edge features MAE	No edge features % acc	No edge features % acc	Edge features % acc	No edge features % ROC-AUC	All models % AP
GCN	$0.469_{\pm 0.002}$		$65.880_{\pm 0.074}$	$54.46_{\pm 0.10}$		$76.06_{\pm 0.97}$ *	$20.20_{\pm 0.24}$ *
GIN	$0.408_{\pm 0.008}$		$85.590_{\pm 0.011}$	$53.28_{\pm 3.70}$		$75.58_{\pm 1.40}$ *	$22.66_{\pm 0.28}$ *
GraphSage	$0.410_{\pm 0.005}$		$50.516_{\pm 0.001}$	$66.08_{\pm 0.24}$			
GAT	$0.463_{\pm 0.002}$		$75.824_{\pm 1.823}$	$65.48_{\pm 0.33}$			
MoNet	$0.407_{\pm 0.007}$		$85.482_{\pm 0.037}$	$53.42_{\pm 0.43}$			
GatedGCN	$0.422_{\pm 0.006}$	$0.363_{\pm 0.009}$	$84.480_{\pm 0.122}$	$69.19_{\pm 0.28}$	$69.37_{\pm 0.48}$		
PNA	$0.320_{\pm 0.032}$	$0.188_{\pm 0.004}$	$86.567_{\pm 0.075}$	$70.46_{\pm 0.44}$	$70.47_{\pm 0.72}$	$79.05_{\pm 1.32}$ *	$28.38_{\pm 0.35}$ *
DGN	**$0.219_{\pm 0.010}$**	**$0.168_{\pm 0.003}$**	**$86.680_{\pm 0.034}$**	**$72.70_{\pm 0.54}$**	**$72.84_{\pm 0.42}$**	**$79.70_{\pm 0.97}$**	**$28.85_{\pm 0.30}$** *

Figure 6. Fine-tuned results of the DGN model against models from (Dwivedi et al., 2020) and (Hu et al., 2020): GCN (Kipf & Welling, 2016), GraphSage (Hamilton et al., 2017), GIN (Xu et al., 2018a), GAT (Veličković et al., 2017), MoNet (Monti et al., 2017), GatedGCN (Bresson & Laurent, 2017) and PNA (Corso et al., 2020). All the models use $\sim 100k$ parameters, except those with * who use $300k$ to $6.5M$. In ZINC the DGN aggregators are $\{$*mean*, dx_1, *max*, *min*$\}$, in PATTERN $\{$*mean*, dx_1, $av_1\}$, in CIFAR10 $\{$*mean*, dx_1, dx_2, *max*$\}$, in MolHIV $\{$*mean*, dx_1, av_1, *max*, *min*$\}$, in MolPCBA $\{$*mean*, *sum*, *max*, $dx_1\}$. Mean and uncertainty are taken over 4 runs for ZINC, PATTERN and CIFAR10 and 10 runs for MolHIV and MolPCBA.

in a wide variety of systems could be expressed as vector fields (spinor, tensor) compatible with the DGN framework, without the need of eigenvectors. One example is magnetic anisotropicity in metals, alloys and organic molecules that is dependant on the relative orientation to the magnetic field. Other examples are the response of materials to high electromagnetic fields; all kind of field propagation in crystals lattices (vibrations, heat, shear and frictional force, young modulus, light refraction, birefringence); multi-body or liquid motion; magnons and solitons in different media, fracture propagation, traffic modelling; developmental biology and embryology, and design of novel materials and constrained structures. Finally applications based on neural operators for ODE/PDE may benefit as well.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. *arXiv:2006.05205 [cs, stat]*, 2020. URL http://arxiv.org/abs/2006.05205.

Antoine, J. P. and Murenzi, R. Two-dimensional directional wavelets and the scale-angle representation. *Signal Processing*, 52(3):259–281, 1996. ISSN 0165-1684. doi: 10.1016/0165-1684(96)00065-5. URL https://www.sciencedirect.com/science/article/pii/0165168496000655.

Barlow, M. T. *Random Walks and Heat Kernels on Graphs*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2017.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. ISSN 1053-5888, 1558-0792. doi: 10.1109/MSP.2017.2693418. URL http://arxiv.org/abs/1611.08097.

Chavel, I. *Eigenvalues in Riemannian geometry*. Academic press, 1984.

Chung, F. and Yau, S. T. Discrete green's functions. *Journal of Combinatorial Theory, Series A*, 91(1):191–

214, 2000. ISSN 0097-3165. doi: 10.1006/jcta.2000.3094. URL http://www.sciencedirect.com/science/article/pii/S0097316500930942.

Chung, F., Graham, F., on Recent Advances in Spectral Graph Theory, C. C., (U.S.), N. S. F., Society, A. M., and of the Mathematical Sciences, C. B. *Spectral Graph Theory*. CBMS Regional Conference Series. Conference Board of the mathematical sciences, 1997. ISBN 9780821803158. URL https://books.google.ca/books?id=4IK8DgAAQBAJ.

Coifman, R. R. and Lafon, S. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2006.04.006. URL https://www.sciencedirect.com/science/article/pii/S1063520306000546. Special Issue: Diffusion Maps and Wavelets.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*, 2020.

Doshi, V. and Eun, D. Y. Fiedler vector approximation via interacting random walks. *arXiv:2002.00283 [math]*, 2000. doi: 10.1145/3379487. URL http://arxiv.org/abs/2002.00283.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Fiedler, M. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 01 1973. doi: 10.21136/CMJ.1973.101168.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR.org, 2017.

Grady, L. J. and Polimeni, J. *Discrete calculus : applied analysis on graphs for computational science*. Springer, 2010.

Grebenkov, D. S. and Nguyen, B.-T. Geometrical structure of laplacian eigenfunctions. *SIAM Review*, 55(4): 601–667, Jan 2013. ISSN 1095-7200. doi: 10.1137/120880173. URL http://dx.doi.org/10.1137/120880173.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Hamilton, W. L. *Graph Representation Learning*. Morgan and Claypool, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Islam, M. A., Jia, S., and Bruce, N. D. B. How much position information do convolutional neural networks encode? *arXiv:2001.08248 [cs]*, 2020. URL http://arxiv.org/abs/2001.08248.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. *arXiv:1802.04364 [cs, stat]*, 2018. URL http://arxiv.org/abs/1802.04364.

Kang, E., Min, J., and Ye, J. C. A deep convolutional neural network using directional wavelets for low-dose x-ray CT reconstruction. *Medical Physics*, 44(10):e360–e375, 2017. ISSN 2473-4209. doi: 10.1002/mp.12344.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. *ICLR2020*, 2019. URL https://openreview.net/forum?id=B1eWbxStPH.

Knyazev, B., Taylor, G. W., and Amer, M. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 4204–4214, 2019.

Kondor, R., Son, H. T., Pan, H., Anderson, B., and Trivedi, S. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

Krizhevsky, A., 2009.

Lanczos, C. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

Levie, R., Monti, F., Bresson, X., and Bronstein, M. M. CayleyNets: Graph convolutional neural networks with complex rational spectral filters. *arXiv:1705.07664 [cs]*, 2018. URL http://arxiv.org/abs/1705.07664.

Levy, B. Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pp. 13–13, 2006. doi: 10.1109/SMI.2006.21.

Luan, S., Zhao, M., Chang, X.-W., and Precup, D. Break the ceiling: Stronger multi-scale deep graph convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 10943–10953, 2019.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2017.

O'Gara, S. and McGuinness, K. Comparing data augmentation strategies for deep image classification. *Session 2: Deep Learning for Computer Vision*, 2019. doi: http://doi.org10.21427/148b-ar75. URL https://arrow.tudublin.ie/impstwo/7.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. An overview of early vision in InceptionV1. *Distill*, 5(4):e00024.002, 2020. ISSN 2476-0757. doi: 10.23915/distill.00024.002. URL https://distill.pub/2020/circuits/early-vision.

Peng, H., Li, J., Gong, Q., Wang, S., Ning, Y., and Yu, P. S. Graph convolutional neural networks via motif-based attention. *arXiv:1811.08270 [cs]*, 2019. URL http://arxiv.org/abs/1811.08270.

Rong, Y., Huang, W., Xu, T., and Huang, J. DropEdge: Towards deep graph convolutional networks on node classification. *ICLR2020*, pp. 17, 2020.

Saerens, M., Fouss, F., Yen, L., and Dupont, P. The principal components analysis of a graph, and its relationships to spectral clustering. In *European conference on machine learning*, pp. 371–383. Springer, 2004.

Sato, R., Yamada, M., and Kashima, H. Approximation ratios of graph neural networks for combinatorial problems. *arXiv preprint arXiv:1905.10261*, 2019.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0197-0. URL https://doi.org/10.1186/s40537-019-0197-0.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. *arXiv:1904.07785 [cs, stat]*, 2019. URL http://arxiv.org/abs/1904.07785.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018a.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462, 2018b.

Yue Lu and Minh N. Do. The finer directional wavelet transform. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 4, pp. 573–576. IEEE, 2005. ISBN 978-0-7803-8874-1. doi: 10.1109/ICASSP.2005.1416073. URL http://ieeexplore.ieee.org/document/1416073/.