GRAPH ANISOTROPIC DIFFUSION

Ahmed A. A. Elhag[*]
African Masters of Machine Intelligence

Gabriele Corso
MIT

Hannes Stärk
MIT

Michael M. Bronstein
University of Oxford / Twitter

ABSTRACT

Traditional Graph Neural Networks (GNNs) rely on message passing, which amounts to permutation-invariant local aggregation of neighbour features. Such a process is isotropic and there is no notion of 'direction' on the graph. We present a new GNN architecture called Graph Anisotropic Diffusion. Our model alternates between linear diffusion, for which a closed-form solution is available, and local anisotropic filters to obtain efficient multi-hop anisotropic kernels. We test our model on two common molecular property prediction benchmarks (ZINC and QM9) and show its competitive performance.

1 INTRODUCTION

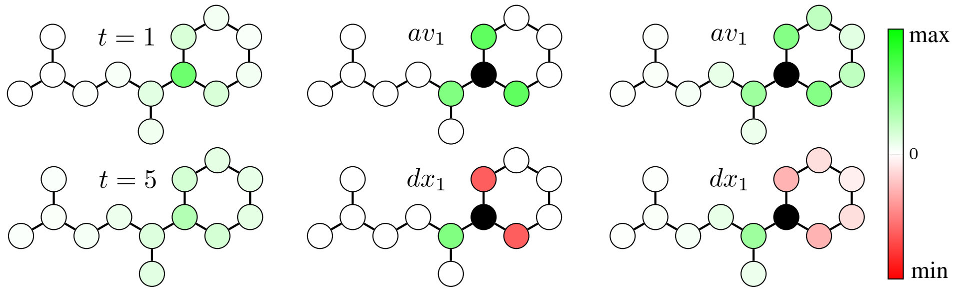
Graphs are a very common mathematical abstraction used to represent complex systems of relations and interactions. However, they present several challenges for machine learning methods because they require algorithms to reason about their underlying structure as well as their node and edge features. Graph Neural Networks (GNNs) are a popular architecture for learning on graph-structured data. GNNs have been successfully applied to problems in different domains such as social networks, recommendation systems (Ying et al., 2018; Fan et al., 2019), chemistry (Gilmer et al., 2017; Sanchez-Lengeling et al., 2019), bioinformatics (Lv et al., 2021; Gainza et al., 2020), and drug discovery (Gaudelet et al., 2021).

The majority of GNNs are based on the *message passing* mechanism (Gilmer et al., 2017), in which the features of each node in the graph are updated by aggregating information from its neighbours. However, recent studies have demonstrated several limitations of message passing, which include limited expressive power (due to equivalence to the Weisfeiler-Lehman graph isomorphism test) (Xu et al., 2019; Corso et al., 2020; Bodnar et al., 2021), as well as the oversmoothing (Oono & Suzuki, 2020) and bottleneck phenomena (Alon & Yahav, 2021).

A recent line of works linked GNNs to diffusion processes on graphs. GRAND (Chamberlain et al., 2021a) and BLEND (Chamberlain et al., 2021b) obtained GNNs from a discretisation of nonlinear and non-Euclidean diffusion PDEs. A more general version of diffusion based on cellular sheaves constructed upon graphs was studied by Bodnar et al. (2022). Klicpera et al. (2019) applied diffusion to the graph structure itself, as a form of graph rewiring that alleviates bottlenecks and improved learning in homophilic settings. Topping et al. (2022) related bottlenecks to graph curvature and used a discrete version of Ricci flow (which, in the continuous setting, can be conceptually interpreted as the 'diffusion of the metric') to improve oversquashing in GNNs.

The lack of canonical ordering of graph nodes forces message passing to use permutation-invariant functions, meaning there is no natural 'direction' on graphs. In geometric graphs and meshes where additional structure is available, learning schemes based on anisotropic diffusion have been studied (Andreux et al., 2014; Boscaini et al., 2016b;a; Sharp et al., 2021). Attempts to define 'pseudodirections' in GNNs using graph motifs were undertaken by Monti et al. (2018). More recently, (Beaini et al., 2021) proposed a scheme for adding directionality in message passing using the graph Laplacian eigenvectors.

[*]Correspondence to aelhag@aimsammi.org



(a) Diffusion for different times t (b) Anisotropic filters: av_1 and dx_1 (c) Anisotropic diffusion kernels

Figure 1: Diffusion kernels and anisotropic filters on a molecular graph. (a) Diffusion for different times t of an initial scalar feature localized at a particular node (the black node in Figures b and c). (b) Local directional smoothing and directional derivative for the black node. (c) Anisotropic diffusion kernels for the black node. Diffusion and anisotropic filter have been applied to a one-hot vector in each node and observed the change in the black node. (The time t is the largest channel time in the first layer of GAD architecture trained on ZINC).

In this work, we build on the recent works of Sharp et al. (2021) and Beaini et al. (2021) to develop anisotropic diffusion on graphs as a basis for GNNs. We propose a new way to perform isotropic diffusion on graphs with learnable kernel size (see Figure 1a) that can be applied efficiently by either solving a linear system or through spectral expansion. We combine this diffusion with a local anisotropic filter (see Figure 1b) to obtain a global anisotropic kernel over the graph (see Figure 1c). This technique is leveraged to propose a novel GNN architecture that we call *Graph Anisotropic Diffusion* (GAD). Empirically, we show that GAD improves the performance of competitive GNNs on two popular molecular property prediction benchmarks: ZINC and QM9.

2 Background

Diffusion equation. Diffusion describes the movement of some quantity from regions of high concentration to lower concentration over time. A classical example of this phenomenon is the diffusion of heat on an iron rod – heat diffuses from warmer parts of the rod to colder ones. Given a domain Ω (e.g., a 1D iron rod), let $x(u,t) : \Omega \times [0,\infty) \to \mathbb{R}$ be a function representing some quantity (e.g., temperature) at position $u \in \Omega$ and time $t \geq 0$. The diffusion process is governed by a partial differential equation (PDE) called the *heat equation*,

$$\frac{\partial}{\partial t} x(u,t) = \frac{\partial^2}{\partial u^2} x(u,t) = \Delta x(u,t), \tag{1}$$

where Δ is the *Laplacian operator* associated with the domain.[1] Assuming initial condition $x(u,0)$, the solution of Equation 1 is given by the heat operator $x(u,t) = \exp(-t\Delta)x(u,0)$, where exp is the operator exponential.

Diffusion equation on surfaces. In computer graphics applications, it is common to work with discretisations of continuous domains (manifolds) such as meshes or point clouds. Let us assume that the domain is sampled at n points, $\{u_1, \ldots, u_n\} \subseteq \Omega$. The sampled function can be represented as an n-dimensional vector $\mathbf{x} = (x(u_1), \ldots, x(u_n))$. The continuous Laplacian operator Δ is discretised using finite elements methods giving rise to the (weak) Laplacian matrix $\mathbf{L}$ and the mass matrix $\mathbf{M}$ (both of size $n \times n$), such that the diffusion rate is approximated by $\mathbf{M}^{-1}\mathbf{L}\mathbf{x}$. Equation 1 becomes

$$\frac{\partial}{\partial t} \mathbf{x}(t) = \mathbf{M}^{-1}\mathbf{L}\mathbf{x}(t), \tag{2}$$

where $\mathbf{x}(t)$ represents the temperature on the nodes of the discretised domain at time t. On meshes, it is customary to use the *cotangent scheme* for Laplacian discretisation and lumped (diagonal) mass matrix, which can be interpreted as a weighted inner product.

[1]On domains with non-Euclidean geometry, the Laplacian is referred to as the Laplace-Beltrami operator.

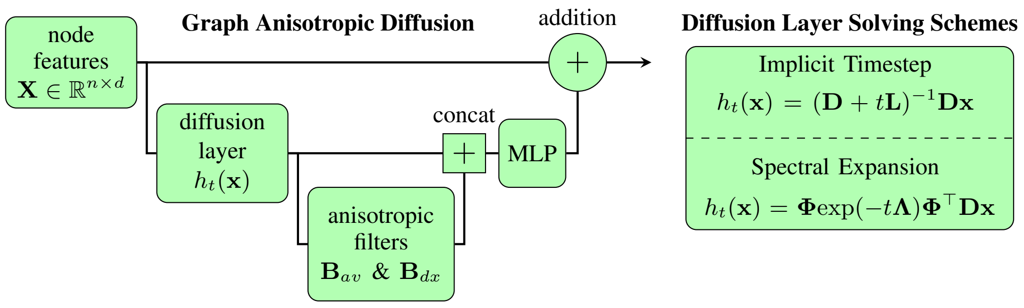

Graph Anisotropic Diffusion
node features $\mathbf{X} \in \mathbb{R}^{n \times d}$
diffusion layer $h_t(\mathbf{x})$
anisotropic filters $\mathbf{B}_{av}$ & $\mathbf{B}_{dx}$
concat
MLP
addition
Diffusion Layer Solving Schemes
Implicit Timestep
$h_t(\mathbf{x}) = (\mathbf{D} + t\mathbf{L})^{-1}\mathbf{D}\mathbf{x}$
Spectral Expansion
$h_t(\mathbf{x}) = \mathbf{\Phi}\exp(-t\mathbf{\Lambda})\mathbf{\Phi}^\top\mathbf{D}\mathbf{x}$

Figure 2: One block of Graph Anisotropic Diffusion.

3 Graph Anisotropic Diffusion

In this section, we present our Graph Anisotropic Diffusion (GAD) method. First, show how a closed-form solution to the graph diffusion equation. Then, we describe the local anisotropic kernels used to provide some notion of direction. Finally, we explain how these two components fit together in GAD. Figure 2 provides a high-level overview of the GAD architecture layer, and Figure 1 provides an intuition behind our construction.

3.1 Learnable Diffusion

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = e$ edges, $\mathbf{A}$ be the $n \times n$ adjacency matrix of $\mathcal{G}$, and $\mathbf{D}$ is the node degree matrix. Our method is based on the discretisation of the diffusion equation (Equation 1) analogous to mesh diffusion equation (Equation 2), where instead we use the graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and the degree matrix as the mass matrix to diffuse the $n \times d$ input feature matrix $\mathbf{X}$:

$$\frac{\partial}{\partial t}\mathbf{X}(t) = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A})\mathbf{X}(t), \tag{3}$$

We use a *channel-wise linear diffusion layer* $h_t : \mathbb{R}^n \to \mathbb{R}^n$, diffusion each channel (column of $\mathbf{X}$) by applying Equation 3 for a time t that is also *learnable* per channel. This enables us to go beyond traditional message passing mechanisms in GNNs. Instead of a single propagation step or discrete random walk, we perform a continuous diffusion with control over the parameter t that determines the kernel size and can be learned end-to-end. As a result, we can optimise the propagation of the features at each node from purely local to completely global by learning each channel's appropriate

How to compute the diffusion? Different numerical schemes are available to solve the linear diffusion equation (Equation 3). To learn and optimise the parameter t via back-propagation, we should confine ourselves to differentiable schemes. In this regard, two simple schemes are used in the experiments: *implicit timestep* and *spectral expansion*. The implementation of both is simple using common scientific libraries. Implicit timestep requires solving a large sparse linear system. In contrast, the spectral expansion scheme has a reduced time complexity and it relies on dense arithmetic but requires some precomputation. In the following we detail both options.

Direct implicit timestep. Sharp et al. (2021) used a single implicit Euler timestep to approximate the diffusion, which in our setting amounts to

$$h_t(\mathbf{x}) = (\mathbf{D} + t\mathbf{L})^{-1}\mathbf{D}\mathbf{x} \tag{4}$$

This approach requires solving a sparse linear system equations for each diffusion operation using Cholesky decomposition.

Spectral expansion scheme. Another way to approximate the solution of the diffusion equation is using spectral (Fourier) expansion. On a graph, an orthogonal Fourier basis is given by the (generalised) eigenvectors of the normalised graph Laplacian, $\mathbf{L}\boldsymbol{\phi}_i = \lambda_i \mathbf{D}\boldsymbol{\phi}_i$. The solution of the diffusion equation can be expressed through the spectral expansion of the heat operator,

$$h_t(\mathbf{x}) \approx \mathbf{\Phi}_k \exp(-t\mathbf{\Lambda}_k)\mathbf{\Phi}_k^\top \mathbf{D}\mathbf{x}, \tag{5}$$

where $\mathbf{\Phi}_k = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k)$ is the $n \times k$ matrix of the first k eigenvectors and $\mathbf{\Lambda}_k = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$ is the $k \times k$ diagonal matrix of the corresponding eigenvalues. This is an approximation due to dropping higher-frequency components, and k plays the role of bandwidth. The matrices $\mathbf{\Phi}_k$ and $\mathbf{\Lambda}_k$ can be pre-computed, as they do not depend on the features $\mathbf{x}$ nor the learnable parameter t. Therefore, with a small approximation error, we obtain a significantly more efficient method than the implicit Euler approach.

3.2 Anisotropic filters

Beaini et al. (2021) proposed the use of anisotropic filters on graphs and defined them as aggregation matrices that allow directional flows between nodes in the graph. These matrices depend on the graph structure and are derived from a vector field $\mathbf{F}$ (represented as an $n \times n$ matrix) that associates each edge with a scalar weight describing the 'flow' from one node to its neighbours. These weights correspond to the magnitude of the flow in a specific direction.

Following Beaini et al. (2021), we use the gradient of the first non-constant eigenvector of the normalised graph Laplacian (the Fiedler vector, which we denote here by $\boldsymbol{\phi}$) to define the vector field

$$\mathbf{F} = \mathbf{A} \odot \nabla \boldsymbol{\phi}, \tag{6}$$

where $\odot$ is element-wise multiplication and $(\nabla \boldsymbol{\phi})_{ij} = \phi_i - \phi_j$ is an $n \times n$ matrix. Beaini et al. (2021) employ this vector field to define two aggregation matrices $\mathbf{B}_{av}$ and $\mathbf{B}_{dx}$ as

$$\mathbf{B}_{av} = |\hat{\mathbf{F}}| \qquad\qquad \mathbf{B}_{dx} = \hat{\mathbf{F}} - \mathrm{diag}(\textstyle\sum_j \hat{\mathbf{F}}_{:,j}), \tag{7}$$

where $\hat{\mathbf{F}}$ denotes the vector field $\mathbf{F}$ after normalising each row by its L_1-norm and $|\cdot|$ denotes the absolute value. Message passing based on these aggregation schemes enables the messages to contain a notion of directionality — an indication of where a message comes from. $\mathbf{B}_{av}$ can intuitively be understood to smooth the signal based on the directions specified by $\mathbf{F}$. In contrast, the aggregation matrix $\mathbf{B}_{dx}$ computes the discrete derivative of the signal. The visualisation of both $\mathbf{B}_{av}$ and $\mathbf{B}_{dx}$ for a specific node in the graph is shown in Figure 1b. We use $\mathbf{B}_{av}$ and $\mathbf{B}_{dx}$ to define the anisotropic filters in our architecture.

3.3 Graph Anisotropic Diffusion Architecture

We combine the diffusion layer in Section 3.1 and the aggregation matrices in Section 3.2 in one architecture which we call Graph Anisotropic Diffusion (GAD). One block of GAD is shown in Figure 2. The input node features are passed through a diffusion layer h_t. Next, we apply a non-linearity to the output of the diffusion and then the aggregation matrices ($\mathbf{B}_{av}$ and $\mathbf{B}_{dx}$). Last, the output of the diffusion layer is concatenated with the output of these aggregation matrices and passed through an MLP. We also use a skip connection in every GAD block, as shown in Figure 2. By combining the linear diffusion and the local anisotropic filters, we can obtain a global anisotropic diffusion, as shown in Figure 1c.

4 Related work

Computer graphics and geometry processing. In computer graphics, the heat kernel of the diffusion equation on a surface is related to its Gaussian curvature. This property was exploited by Sun et al. (2009); Bronstein & Kokkinos (2010) to construct heat kernel signature (HKS) shape descriptors. Andreux et al. (2014) introduced the use of the anisotropic Laplace-Beltrami operator to give a directional diffusion operation on discrete meshes. Boscaini et al. (2016b) also proposed a convolutional neural network architecture based on anisotropic diffusion kernels to learn an intrinsic dense correspondence between non-rigid shapes. More recently, Sharp et al. (2021) combined directional filters with diffusion for learning on meshes and point clouds.

Differential equations on graphs. Diffusion processes and random walks on graphs have long been exploited, including the classical PageRank algorithm (Page et al., 1999) (steady state of diffusion on the web graph) that powered the early version of Google search engine ranking, or non-linear dimensionality reduction by diffusion maps proposed by Coifman & Lafon (2006). In the graph ML literature, GNNs were formulated as ordinary differential equations (ODEs) by Avelar et al. (2019);

Poli et al. (2021), continuous diffusion-type processes (Xhonneux et al., 2020; Gu et al., 2020), or physically-informed architectures (Sanchez-Gonzalez et al., 2019). On the other hand, solving the non-Euclidean diffusion equations using the Laplacian eigenvectors and eigenvalues was considered a crucial tool for some early techniques on graph neural networks (Henaff et al., 2015; Kipf & Welling, 2017; Levie et al., 2018). GRAND (Chamberlain et al., 2021a) formulated attention-type GNNs as discretisations of the diffusion equation. BLEND (Chamberlain et al., 2021b) proposes a non-Euclidean extension to GRAND. The key difference of our work from GRAND and BLEND (Chamberlain et al., 2021a;b) is that we used closed-form solution to graph diffusion, interleaved with an anisotropic filter, rather than a numerical solver.

5 Experiments and results

To evaluate our model, we tested it on two popular benchmarks: ZINC (Irwin et al., 2012) and coordinate-free QM9 (Ramakrishnan et al., 2014). The implementation using PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019) libraries is publicly available at https://github.com/Ahmed-A-A-Elhag/GAD. Our results on each dataset are presented with a comparison with previous models in Tables 1 and 2. (For QM9, we test our model on the first seven properties provided by the PyTorch Geometric (Fey & Lenssen, 2019) library). Appendix A contains the implementation details and hyperparameter settings for each dataset.

Table 1: Mean absolute error (MAE) for ZINC. GAD-i uses the implicit timestep approach and GAD-s the spectral expansion scheme. *Uses additional tailored features such as cycles.

Model	No edge features	Edge features
GCN	0.469± 0.002	-
GIN	0.408± 0.008	-
GraphSage	0.410± 0.005	-
GAT	0.463± 0.002	-
MoNet	0.407± 0.007	-
GatedGCN	0.422± 0.006	0.363 ± 0.009
PNA	0.320± 0.032	0.188± 0.004
DGN	0.219± 0.010	0.168± 0.003
HIMP*	-	0.151± 0.006
SMP*	0.219±	0.138±
GSN*	0.140± 0.006	0.115± 0.012
GAD-i	0.181± 0.004	0.139± 0.007
GAD-s	0.194± 0.006	0.140± 0.004

Table 2: Mean absolute error (MAE) for QM9's properties.

	Unit	PNA	**DGN**	**GAD-s**
μ	D	0.365	0.354	0.338
α	a_0^3	0.255	0.250	0.240
ϵ_{HOMO}	meV	71.5	71.1	64.3
ϵ_{LUMO}	meV	70.1	68	63
$\Delta\epsilon$	meV	100.2	99.2	96.8
$\langle R^2 \rangle$	${a_0}^2$	18.3	17.2	16.5
ZPVE	meV	6.57	6.03	4.84

Given that DGN (Beaini et al., 2021) is the baseline GNN that our model was constructed upon, we consider this as the main comparison point to analyse the effectiveness of the diffusion process presented. On ZINC, the addition of the diffusion layer reduces the MAE by more than 17%, and on QM9, it improves the performance in every single property. Note that while methods using the addition of tailored features or message passing structure achieve better performances on ZINC, these improvements are orthogonal to the ones proposed in this work and could be combined.

6 Conclusion

We proposed Graph Anisotropic Diffusion (GAD), a novel method that consists of a new efficient way of performing isotropic diffusion on graphs combined with local anisotropic kernels. This approach is used to obtain a global anisotropic diffusion. Empirical results on popular molecular property prediction benchmarks confirm the improved predictive capacity of our method. We hope this work spurs future research on alternative methods to obtain anisotropic diffusion on graphs and theoretical analyses of their expressive power.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *ICLR*, 2021.

Mathieu Andreux, Emanuele Rodola, Mathieu Aubry, and Daniel Cremers. Anisotropic laplace-beltrami operators for shape analysis. In *ECCV*, 2014.

Pedro H. C. Avelar, Anderson R. Tavares, Marco Gori, and Luis C. Lamb. Discrete and continuous deep residual learning over graphs. *arXiv preprint*, 2019.

Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In *ICML*, 2021.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F. Montúfar, Pietro Lió, and Michael M. Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *ICML*, 2021.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *arXiv preprint*, 2022.

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *arXiv preprint*, 2016a.

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Michael M Bronstein, and Daniel Cremers. Anisotropic diffusion descriptors. *Computer Graphics Forum*, 35(2):431–441, 2016b.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint*, 2021.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint*, 2018.

Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation. In *ICML*, 2020.

Michael M. Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *CVPR*, pp. 1704–1711, 2010.

Ben Chamberlain, James Rowbottom, Maria Gorinova, Michael M. Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: graph neural diffusion. In *ICML*, 2021a.

Benjamin Paul Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael M Bronstein. Beltrami flow and neural diffusion on graphs. *arXiv preprint*, 2021b.

Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *NeurIPS*, 2020.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint*, 2020.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *International Conference on World Wide Web*, 2019.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint*, 2019.

Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint*, 2020.

P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. M. Bronstein, and B. E. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nat Methods 17*, 52(7):184–192, 2020.

Thomas Gaudelet, Ben Day, Arian R. Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy B. R. Hayter, Richard Vickers, Charles Roberts, Jian Tang, David Roblin, Tom L. Blundell, Michael M. Bronstein, and Jake P. Taylor-King. Utilising graph machine learning within drug discovery and development. *arXiv preprint*, 2021.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. In *NeurIPS*, 2020.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint*, 2015.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52 (7):1757–1768, 2012.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *NeurIPS*, 2019.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *arXiv preprint*, 2018.

Guofeng Lv, Zhiqiang Hu, Yanguang Bi, and Shaoting Zhang. Learning unknown from correlations: Graph neural network for inter-novel-protein interaction prediction. *arXiv preprint*, 2021.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pp. 5425–5434, 2017.

Federico Monti, Karl Otness, and Michael M Bronstein. Motifnet: a motif-based graph convolutional network for directed graphs. In *IEEE Data Science Workshop (DSW)*, pp. 225–228, 2018.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, 1999.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint*, 2021.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):1–7, 2014.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint*, 2019.

Benjamin Sanchez-Lengeling, Jennifer N. Wei, Brian K. Lee, Richard C. Gerkin, Alán Aspuru-Guzik, and Alexander B. Wiltschko. Machine learning for scent: Learning generalizable perceptual representations of small molecules. *arXiv preprint*, 2019.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *arXiv preprint*, 2021.

Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proceedings of the Symposium on Geometry Processing*, pp. 1383–1392, 2009.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*, 2022.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Clément Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. In *NeurIPS*, 2020.

Louis-Pascal A. C. Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *ICML*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

A IMPLEMENTATION DETAILS AND HYPERPARAMETER SETTINGS

Following Beaini et al. (2021), in our implementation of the local filters use mean, max, and min aggregators on top of $\mathbf{B}_{av}$ and $\mathbf{B}_{dx}$, followed by the degree scalers of PNA (Corso et al., 2020). Below we describe each dataset, including implementation details and hyperparameter settings.

A.1 ZINC

ZINC is a molecule graphs dataset in which the task is to predict the penalised water-octanol partition coefficient of a certain molecule (formulated as a graph regression task). We use the splits and evaluation criteria provided by Dwivedi et al. (2020): 10K training, 1K validation, and 1K test, with a maximum of 100K parameters and mean absolute error (MAE) as evaluation metric. We compare our model against various GNNs architectures: GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019), GraphSage (Hamilton et al., 2017), GAT (Velickovic et al., 2018), MoNet (Monti et al., 2017), GatedGCN (Bresson & Laurent, 2018), PNA (Corso et al., 2020), DGN (Beaini et al., 2021), HIMP (Fey et al., 2020), SMP (Vignac et al., 2020), and GSN (Bouritsas et al., 2021). We ran each solving scheme (implicit timestep and spectral expansion) five times to estimate the mean performance and its standard deviation (Table 1).

Hyperparameter settings. We use the same hyperparameter settings in DGN (Beaini et al., 2021), and decrease the hidden dimensions to keep the same number of parameters. In the spectral expansion scheme, we use the number of eigenvectors $k \in \{20, 25, 30\}$.

A.2 QM9

QM9 is a common molecule graph benchmark, which provides different quantum mechanical properties of small molecules. we test our model on the first seven properties provided by the PyTorch Geometric (Fey & Lenssen, 2019) library (Table 2) keeping again constant the number of parameters and removing the 3D coordinates from the model's inputs. For the data split, we use around 110k training, 10k validation, and 10k test samples as provided by Brockschmidt (2020). The evaluation metric is mean absolute error (MAE).

Hyperparameter settings. We fine-tuned GAD over the following hyperparameter settings:

- Weight decay $\in \{3 \times 10^{-3}, 3 \times 10^{-4}, 3 \times 10^{-5}, 3 \times 10^{-6}, 3 \times 10^{-7}\}$.
- Dropout $\in \{0, 0.1, 0.2, 0.3\}$.
- $lr \in \{1 \times 10^{-3}\}$.
- Batch size $\in \{128, 256\}$.
- Number of layers $\in \{6, 7, 8, 9, 10\}$.
- Hidden dimensions $\in \{75, 90, 130, 150\}$.
- Number of eigenvectors $k \in \{15, 20, 25\}$.
- Aggregators $\in \{(mean, av_1, dx_1), (mean, max, min, dx_1), (mean, max, min, av_1, dx_1), (mean, sum, max, dx_1)\}$.

For PNA (Corso et al., 2020) and DGN (Beaini et al., 2021) models, we ran each one with the hyperparameters obtained from the search above and increased the size of hidden dimensions to have the same number of parameters. Aggregators used in PNA (Corso et al., 2020) are $(mean, max, min, std)$, and in DGN (Beaini et al., 2021) are $(mean, sum, max, dx_1)$.