Dirac–Bianconi Graph Neural Networks – Enabling Non-Diffusive Long-Range Graph Predictions

Christian Nauck [1] **Rohan Gorantla** [2 3] **Michael Lindner** [1] **Konstantin Schürholt** [4] **Antonia S. J. S. Mey** [2] **Frank Hellmann** [1]

Editors: *S. Vadgama, E.J. Bekkers, A. Pouplin, S.O. Kaba, H. Lawrence, R. Walters, T. Emerson, H. Kvinge, J.M. Tomczak, S. Jegelka*

Abstract

The geometry of a graph is encoded in dynamical processes on the graph. Many graph neural network (GNN) architectures are inspired by such dynamical systems, typically based on the graph Laplacian. Here, we introduce Dirac–Bianconi GNNs (DBGNNs), which are based on the topological Dirac equation recently proposed by Bianconi. Based on the graph Laplacian, we demonstrate that DBGNNs explore the geometry of the graph in a fundamentally different way than conventional message passing neural networks (MPNNs). While regular MPNNs propagate features diffusively, analogous to the heat equation, DBGNNs allow for coherent long-range propagation. Experimental results showcase the superior performance of DBGNNs over existing conventional MPNNs for long-range predictions of power grid stability and peptide properties. This study highlights the effectiveness of DBGNNs in capturing intricate graph dynamics, providing notable advancements in GNN architectures.

1. Introduction

Understanding the geometric and physical properties of real-world data is crucial for AI applications in physics, chemistry, and robotics. For the common case of graph structured data, various graph neural network (GNN) architectures have been developed (Wu et al., 2021).

GNNs have been motivated in various ways. Spectral approaches see the eigenbasis of the graph Laplacian $\boldsymbol{L}$ as playing a similar role as that of Fourier modes in spatial data.

Spatial convolution approaches, instead, build on the idea of aggregating features by considering spatial neighborhoods. Message passing is one of the most prominent implementations of this idea. For the edge connecting nodes i and j, we construct a message e_{ij} from inputs x_i^{in}, x_j^{in} at its ends, and some invariant edge features e_{ij}^f. Then each node aggregates messages on the edges connected to it to produce an output. Denoting general, potentially non-linear, functions by f, g, h, and for the set of neighbors $\mathcal{N}_i$ of node i, the most general structure usually assumed for message passing neural networks (MPNNs) is:

$$e_{ij} = g\left(x_i^{\text{in}}, x_j^{\text{in}}, e_{ij}^f\right), \tag{1}$$

$$x_i^{\text{out}} = f\left(x_i^{\text{in}}, \sum_{j\in\mathcal{N}_i} h(e_{ij}, x_j^{\text{in}})\right). \tag{2}$$

Here, g can compute simple differences across edges and attention scores. There is a large body of work exploring the potential of this style of architecture, including variations of updates to edge features. In this context, the case where edge features are updated simultaneously with node features from layer to layer (see for example Yang & Li (2020); Chen & Chen (2021); He et al. (2023); Zhou et al. (2023)) has not been explored as extensively:

$$e_{ij}^{\text{out}} = g\left(x_i^{\text{in}}, x_j^{\text{in}}, e_{ij}^{\text{in}}\right), \tag{3}$$

$$x_i^{\text{out}} = f\left(x_i^{\text{in}}, \sum_{j\in\mathcal{N}_i} h(e_{ij}^{\text{in}}, x_j^{\text{in}})\right). \tag{4}$$

In contrast to ordinary MPNNs, the variable e cannot be eliminated from the equations. Edge features propagate

[1]Potsdam Institute for Climate Impact Research, Telegrafenberg A31, 14473 Potsdam, Germany [2]EaStCHEM School of Chemistry, University of Edinburgh, EH9 3FJ, UK [3]School of Informatics, University of Edinburgh, EH8 9AB, UK [4]AIML Lab, University of St. Gallen, Rosenbergstrasse 30, CH-9000 St. Gallen, Switzerland. Correspondence to: Christian Nauck <nauck@pik-potsdam.de>, Frank Hellmann <hellmann@pik-potsdam.de>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at the 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number 251, 2024. Copyright 2024 by the author(s).

from layer to layer. Note that this also falls outside the edge update model of Battaglia et al. (2018), which considers node updates of the form $\sum h(e_{ij}^{\text{out}}, x_j^{\text{in}})$.

Exploring GNNs as dynamical systems These structures resemble that of a discrete dynamical system on a network. This perspective has led to novel approaches to GNN architectures that draw inspiration from dynamical systems to overcome limitations observed in earlier architectures. A notable example is the work of Rusch et al. (2022), who introduce neural networks based on inertial Kuramoto oscillator networks (Kuramoto, 1975; Acebrón et al., 2005; Rodrigues et al., 2016). They argue that oversmoothing – the observation that many GNN architectures tend to average out features across the graph when iterated too deeply – is analogous to synchronization in such dynamical systems. The fact that the synchronous manifold is unstable for oscillator dynamics under a natural parameter condition then suggests that its architecture should not suffer from oversmoothing.

GNN architectures have been inspired by theoretical physics in numerous related works. Notable works draw from geometric curvature (Topping et al., 2022), discrete dynamical systems (Oono & Suzuki, 2021), ordinary differential equations (Poli et al., 2021), as well as partial differential equations and their discretization schemes (Chamberlain et al., 2021b;a; Eliasof et al., 2021).

Generally, these works are based on dynamical systems defined in terms of variants of the graph Laplacian. The paradigmatic example of a dynamical system defined by the Laplacian is the heat equation, which describes diffusion of a density ρ in space (r) and time (t):

$$\partial_t \rho(r,t) = -\Delta \rho(r,t). \tag{5}$$

The process of heat spreading involves the gradual equalization of temperature throughout the system. This spreading can be adjusted in various ways, leading to the development of analogous GNN architectures. Advection plays a role in moving heat around (Eliasof et al., 2023; Zhao et al., 2023), different regions may exhibit varying heat conductivities, and anomalous diffusion can impact the speed at which the system reaches equilibrium (Maskey et al., 2023).

The main motivation of our work is to investigate a new class of physical processes that can be used as the basis for GNN architectures. Consider the way that light travels: It can slow down in a medium, reflect, refract, and travel long distances without losing its shape. The underlying equation has oscillating plane waves as solutions, which can be combined into wave packets that maintain their shape while traveling long distances. Fiber optics provide a great example of how a medium can shape the path of a signal while the wave packet keeps its shape, allowing a light pulse to transmit information over hundreds of kilometers. Unfortunately, there is no direct analogy to electromagnetic equations in graphs, since there is no natural gradient operator.

Inspiration from the Dirac equation on networks Instead, we base our work on the topological Dirac equation on networks, recently introduced by Bianconi (2021). The Dirac equation is one of the fundamental equations of quantum mechanics. It describes the evolution of the Dirac field, representing most elementary particles such as electrons, protons, and quarks. Like electromagnetic equations, it allows wave packets to maintain their shape.

The Dirac equation is based on the Dirac operator, a square root of the Laplacian. It is obtained by mixing spatial derivatives with transformations in 'internal space', i.e., transformations between different components of the Dirac field. Bianconi (2021) builds on earlier work on quantum information processing by Lloyd et al. (2016), introducing a Dirac operator on simplicial complexes to create a topological Dirac equation for graphs.

The topological Dirac equation falls outside the dynamics captured by the standard MPNN framework shown in Equation (1), as it treats edges and nodes on the same footing. It is thus an example of the less studied layer of the type described by Equation (3) with propagating edge features. Given that it is based on a physical equation that allows for wave packets to propagate long distances without losing their shape, we expect this architecture to be of interest where oversmoothing and long distance dependencies are important. Furthermore, the fact that edges and nodes are treated equally is also appropriate for some tasks.

Potential for longe-range predictions One motivating application for GNNs with long-range capabilities inspired by dynamical systems is predicting stability properties of power grids. Ringsquandl et al. (2021) show that GNNs with 13 or more layers are needed to achieve good performance in predicting stability properties of power grids with roughly 300 buses. This differs from many commonly used benchmark datasets where GNNs with only 2-3 layers perform best. This suggests that architectures that tend to oversmooth will struggle in the context of power grids. Power grids have node and edge features of similar nature and importance, suggesting the need to treat edges and nodes similarly, rather than considering the former as just a coupling for the latter. In addition, the complex topological aspects of the graph topology play a significant role in shaping the dynamic properties in power grids. A striking example is shown by Nitzbon et al. (2017), who find that certain desynchronization modes of their power grid model occur only in specific topological settings, regardless of the node features.

Our key contributions are:

- We analyze the diffusive character of conventional MPNNs by visualizing the trajectories of signals through the GNNs.

- We generalize the topological Dirac equation and use it to define the novel Dirac–Bianconi T-Step Layer (DBTS).
- We show experimentally that this layer allows for shape-preserving long-range propagation of feature activation along the graph. MPNN architectures with the same random weights do not exhibit this behavior. We also show empirically that the Dirichlet energy does not go to zero under repeated applications of this layer.
- To validate its performance on benchmark datasets, we use an architecture with several DBTS layers and skip connections. We find that the new layer shows superior performance on challenging power grid tasks with crucial long-range dependencies. On molecular tasks for predicting peptide properties from long-range benchmark datasets, DBGNN outperforms conventional message passing (MPNN) methods while using a quarter of the parameters, and is competitive with transformer-based GNNs.

Our work presents a key contribution to improving GNNs and providing long-range capabilities by propagating node and edge features. This paves the way for assembling more complex graph-based datasets that rely on edge features and long-range propagation. In the following sections, we introduce Bianconi's Dirac operator for graphs, as well as the topological Dirac equation and our generalization thereof in more detail. This provides the basis for the introduction of the DBGNN layer. We then present experimental results and compare the performance with benchmark models.

2. Background

Notation: Graphs $\mathcal{G}$ consist of nodes $\mathcal{N}$ and edges $\mathcal{E}$. Each edge occurs twice, with the two possible orientations, and we write an edge e as an ordered pair $[i, j] \in \mathcal{E}$ of nodes $i, j \in \mathcal{N}$. The set of neighbors of node i is denoted as $\mathcal{N}_i$. The space of features on an edge/node is called $F_{e/n}$, the space of all edge features of our graph $\mathcal{G}$ is $F_e^{\mathcal{E}}$, and $F_n^{\mathcal{N}}$ for node features.

The structure of a graph can be described using the incidence matrix $\boldsymbol{B} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{E}|}$:

$$B_{ie} = \begin{cases} +1 \text{ if } e = [i, j] \\ -1 \text{ if } e = [j, i] \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

Introducing the Dirac operator: The incidence matrix $\boldsymbol{B}$ maps from the edge space to the node space. It can be checked that matrix $\boldsymbol{B}\boldsymbol{B}^\dagger$ gives the usual Laplacian matrix, where † denotes the conjugate transpose. Consider a vector of node features x and a vector of edge features e. Bianconi's Dirac operator on graphs is given by:

$$\partial_{DB} \begin{pmatrix} x \\ e \end{pmatrix} = \begin{pmatrix} 0 & b\boldsymbol{B} \\ (b\boldsymbol{B})^\dagger & 0 \end{pmatrix} \begin{pmatrix} x \\ e \end{pmatrix} = \begin{pmatrix} b\boldsymbol{B}e \\ b^*\boldsymbol{B}^\dagger x \end{pmatrix} \tag{7}$$

for some $b \in \mathbb{C}$. We see that this operator maps node features to edge features, and vice versa. The equation considered by Bianconi (2021) contains a mass term β. Taking x and e as a function of $t \in \mathbb{R}$, the topological Dirac equation is

$$i\partial_t \begin{pmatrix} x(t) \\ e(t) \end{pmatrix} = \left(\partial_{DB} + \begin{pmatrix} \beta & 0 \\ 0 & -\beta \end{pmatrix}\right) \begin{pmatrix} x(t) \\ e(t) \end{pmatrix} \tag{8}$$

$$= (\partial_{DB} + \boldsymbol{W}_\beta) \begin{pmatrix} x(t) \\ e(t) \end{pmatrix} . \tag{9}$$

Bianconi (2021) discusses in detail how this equation relates to the motivating Dirac equation for certain graphs; however, it is not at all obvious what the analog of plane waves and wave packets should be for arbitrary topologies. This is a topic of ongoing research.

At the same time, the spectral properties of this equation make it clear that it is fundamentally different from the heat equation. Recall that the heat equation is $\partial_t x = \boldsymbol{\Delta} x$ with solutions given in terms of eigenvectors v_i and eigenvalues λ_i by $\sum_i e^{\lambda_i t} v_i$. Since $\boldsymbol{\Delta}$ is a negative semi-definite matrix, we have $\lambda_i \leq 0$. For connected graphs, only one λ is zero and all solutions decay to the corresponding eigenvector.

In contrast, the operator $\partial_{DB} + \boldsymbol{W}_\beta$ has an equal number of positive and negative eigenvalues (Bianconi, 2021). These are bounded away from zero by $|\beta|$. Thus, the evolution according to this operator cannot equilibrate – there are always expanding and contracting directions. Furthermore, if we note the imaginary unit in front of the time-derivative on the left-hand side, $i\partial_t$, the positive and negative eigenvalues correspond to rotating and counter-rotating oscillations in time, which can be superimposed to approximate temporal envelopes for wave packets. Spatially, these oscillations couple the node and edge spaces together. We therefore hypothesize that such wave packets propagate directly into the graph.

Details on the Dirac operator: For completeness, the following subsection discusses further aspects and motivation for the Dirac operator, although the details are not necessary for understanding the rest of the paper.

In differential geometry, any square root of the Laplacian (typically on a vector bundle over a Riemannian manifold) is called a Dirac operator. While the eigenvalues of the Laplacian operator show how diffusive processes disperse, they do not distinguish between different directions in the manifold. In contrast, those of the Dirac operators typically do so by coupling directions in space to different directions in

the bundle. This can be illustrated by the simplest example. For the tangent bundle over $\mathbb{R}$, the Dirac operator $-i\partial_x$ has eigenvalues ± 1 for eigenmodes $e^{\pm it}$. The Laplace operator has the same eigenmodes, but both correspond to the eigenvalue 1. Exponentiating the Laplacian leads to the heat kernel, which smooths out the differences and leads to equilibration in the long run: All states converge to the kernel of the Laplace operator. In contrast, exponentiating $-i\partial_x$ simply induces shifts along the real axis: $e^{-si\partial_x} f(x) = f(x-s)$, which can be verified by taking the Fourier transform.

As already noted above, the incidence matrix $\boldsymbol{B}$ satisfies $\boldsymbol{L} = \boldsymbol{B}\boldsymbol{B}^\dagger$. In the context of homological algebra, $\boldsymbol{B}$ is called the boundary operator, and $\boldsymbol{B}^\dagger\boldsymbol{B}$ is the so-called one-down Laplacian connecting edges to edges. Such boundary operators play a central role in the extension of GNNs to simplicial complexes. A principled approach to simplicial message passing using this approach is given in Bodnar et al. (2021).

In our context, the relationship between the incidence matrix and the Laplacian suggests that the incidence matrix should play a role similar to that of the Dirac operator. However, since it maps between edges and nodes, it cannot be used directly to update features, which motivates the definition above.

As discussed in detail in (Bianconi, 2021), this Dirac operator does not capture directionality in the same way as the usual Dirac operators of quantum mechanics. For example, on a regular lattice, the operator does not distinguish between edges that are parallel or orthogonal to each other. However, it does mix edge features and node features in nontrivial ways. In addition, its eigenstates encode topological features of the graph on both the edges and the nodes.

Since ∂_{DB} squares the block matrix consisting of the usual graph Laplacian and the one-down Laplacian, iteration of this operator does not introduce any interesting non-diffusive dynamics on the feature space of the graph. However, as the spectral analysis shows, the dynamical equations based on it are genuinely different. We will see further evidence of this difference in the numerical experiments that study the spreading behavior of feature activation.

3. The Dirac–Bianconi Layer

We consider the Euler discretization of Equation (9) and introduce higher-dimensional feature spaces of dimension d_n for nodes and dimension d_e for edges, $F_n = \mathbb{R}^{d_n}$ and $F_e = \mathbb{R}^{d_e}$. Consider coupling matrices $\boldsymbol{W}^{ne} \in \mathbb{R}^{d_n \times d_e}$ and $\boldsymbol{W}^{en} \in \mathbb{R}^{d_e \times d_n}$, and mass matrices $\boldsymbol{W}^n_\beta \in \mathbb{R}^{d_n \times d_n}$, $\boldsymbol{W}^e_\beta \in \mathbb{R}^{d_e \times d_e}$, we can write

$$x_i(t+1) = x_i(t) + \boldsymbol{W}^{ne} \sum_{j \in \mathcal{N}_i} e_{ij}(t) + \boldsymbol{W}^n_\beta x_i(t), \quad (10)$$
$$e_{ij}(t+1) = e_{ij}(t) + \boldsymbol{W}^{en}(x_i(t) - x_j(t)) - \boldsymbol{W}^e_\beta e_{ij}(t).$$

We call this equation the generalized linear Dirac–Bianconi equation. Here $F_n^{\mathcal{N}} = \mathbb{R}^{d_n|\mathcal{N}|}$ denotes the total node feature space and $F_e^{\mathcal{E}}$ the total edge feature space. However, after applying ∂_{DB} we also have the space of one edge feature space per node, $F_e^{\mathcal{N}}$, and one node feature space per edge, $F_n^{\mathcal{E}}$. The action of $\boldsymbol{W}^{ne}$ and $\boldsymbol{W}^{en}$ then maps us back to the original $F_n^{\mathcal{N}}$ and $F_e^{\mathcal{E}}$. This equation contains both the wave behavior and the expanding/contracting dynamics as a special case. The oscillatory behavior occurs when we mimic the imaginary unit by making the right-hand side of Equation (10) antisymmetric: $\beta_{n/e} = -\beta^\dagger_{n/e}$, and $\boldsymbol{W}^{ne} = -\boldsymbol{W}^{en\dagger}$. This should be compared to a simple linear message passing neural network (MPNN)-style dynamics with edge weights, omitting activation functions:

$$x_i(t+1) = x_i(t) + \boldsymbol{W}^{\text{MPNN}}_n \sum_{j \in \mathcal{N}_i} e_{ij}(t) + \beta_n x_i(t), \quad (11)$$
$$e_{ij}(t) = \boldsymbol{W}^{\text{MPNN}}_e (x_i(t) - x_j(t))$$

while the edge messages change over time, there is no time evolution of the edge features themselves. The messages can be trivially eliminated, while this is not the case for Equation (10). This changes the way dynamics spread in the network, see Figure 1. The propagation of edge features allows oscillations to couple directly to the graph structure, enabling waves to propagate. Note that this is true even if the dataset itself has no edge features.

Equation (10) is of the general form considered in Bodnar et al. (2021), but the specific form here was not considered there. From the perspective of simplicial topology, the main difference is that we use the boundary and coboundary operators, together with a simultaneous update of nodes and edges, but no Laplacian. For simplicial two-complexes this has been considered (Bunch et al., 2020), but we are not aware of any previous work of this kind on graphs.

To conclude this section, Laplacian operators, which are common message passing frameworks, lead to smoothing with loss of information. In contrast, the Dirac operator, with its form of the wave equation, does not lead to oversmoothing, which can also be seen visually in Figure 1.

4. Dirac–Bianconi Graph Neural Networks

In the following, we use the generalized linear Dirac–Bianconi equation (Equation (10)) to define a novel GNN layer for problems where long-range interactions in the graph are expected to play a profound role. This is achieved

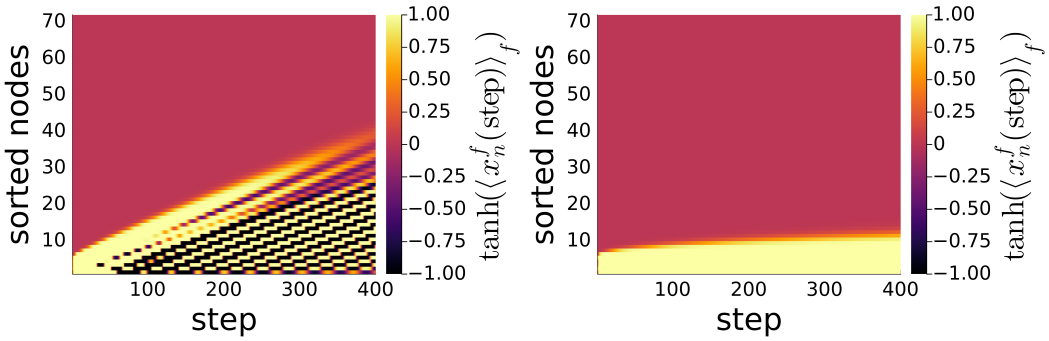


Figure 1. Feature activation versus steps of linear DB Equation (10) (left) and MPNN Equation (11) (right) on a path graph, $d_n = d_f = 1$, same random weights. The initial condition has all features zero, except at node 1, where the node features are randomly activated. Linear DB shows activation moving linearly down the graph, while MPNN shows diffusion.

by explicitly considering node and edge features in a coupled system and by using the Dirac operator, which does not smooth wave signals.

We define the DB 1-Step layer as one step of the linear DB Equation (10), followed by a dropout and a non-linearity. The matrices $\boldsymbol{W}^{ne} \in \mathbb{R}^{d_n \times d_e}$, $\boldsymbol{W}^{en} \in \mathbb{R}^{d_e \times d_n}$, $\boldsymbol{W}^n_\beta \in \mathbb{R}^{d_n \times d_n}$, and $\boldsymbol{W}^e_\beta \in \mathbb{R}^{d_e \times d_e}$ are learnable weights. This layer is sketched in Figure 2a. We sequentially apply T such layers with shared weights to obtain the DB T-step layer of Figure 2b. The full Dirac Bianconi Graph Neural Network (DBGNN) with K T-step layers is then constructed as in Figure 2c:

- First we linearly map the input features to the hidden feature spaces $F_n = \mathbb{R}^{d_n^{\text{hidden}}}$ and $F_e = \mathbb{R}^{d_e^{\text{hidden}}}$ for nodes and edges, respectively.
- We then alternate DB T-step layers and skip connections that mix in the input features using a linear map K times. This allows for different dynamics that see both the initial conditions and the features processed by the previous layers.
- Finally, we use MLP to map from the hidden feature dimension to the output dimension, optionally followed by pooling and another MLP layer.

Such a DBGNN makes $KT/2$ node-to-node hops on the graph. Before testing DBGNNs on real datasets, we perform synthetic analyses that demonstrate the key properties, including long-range capabilities.

5. Synthetic analysis of DBGNN properties

DBGNNs do not suffer from oversmoothing A common way to understand whether an architecture suffers from oversmoothing is to study the different variations of the Dirichlet energy (DE) (Zhou et al., 2021; Wang et al., 2023; Rusch et al., 2022; Chen et al., 2023; Liu et al., 2023; Fu et al., 2023; Di Giovanni et al., 2023), for which the definition and computation are given in Appendix A.2.

To assess the intrinsic equilibration properties of DBGNNs in comparison to Graph Convolutional Networks (GCNs), we evaluate the DE over approximately 1,000 steps for DBGNNs and 100 steps for GCNs in untrained networks. GCNs are included in our analysis due to their widespread use and established effectiveness in various applications.

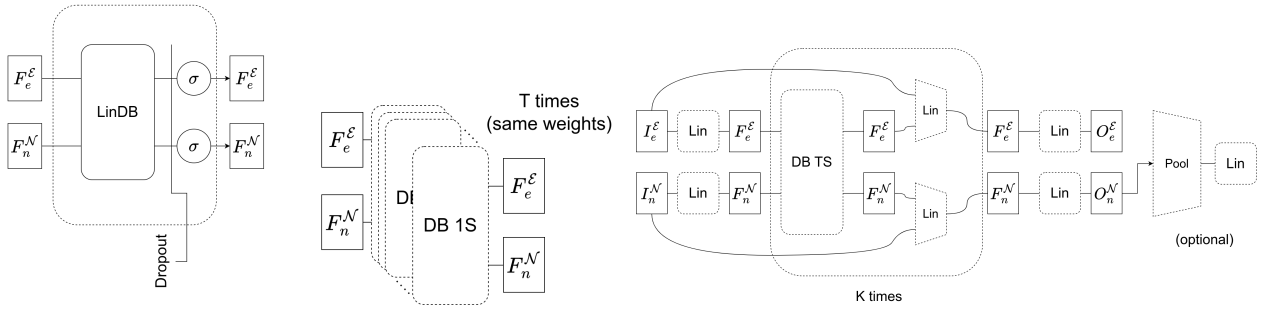
The results are shown in Figure 3. From the spectral analysis of Equation (9), we find that no equilibration occurs for DBGNNs, while GCNs introduced by Kipf & Welling (2017) quickly lose heterogeneity. Hence, DBGNNs allow information to be propagated across very long ranges, while in the case of GCNs, information is diffused quickly.

Wave aspects of DBGNNs enable deep propagation of signals To analyze the long-range capabilities of DBGNNs, we analyze how the layers can spread a localized feature into a graph. Compared to MPNNs, there are two aspects that might enable deep spreading: One is the intrinsic wave dynamics of the linear DB equation, and the other is that we apply a non-linearity to the edges. This second aspect resembles the approach of (Bodnar et al., 2022; Zhao et al., 2023; Rossi et al., 2023). It is noteworthy that with a ReLU activation function on the edges, either $x_i - x_j$ or $x_j - x_i$ is completely suppressed. This induces a strong directionality in the behavior of the layer, which could enable long-range propagation.

To investigate the effect of edge non-linearity and edge updating separately, we will compare Equation (10) and the iterated DB 1-step layer with Equation (11) and an MPNN Equation (12) with and without edge non-linearity:

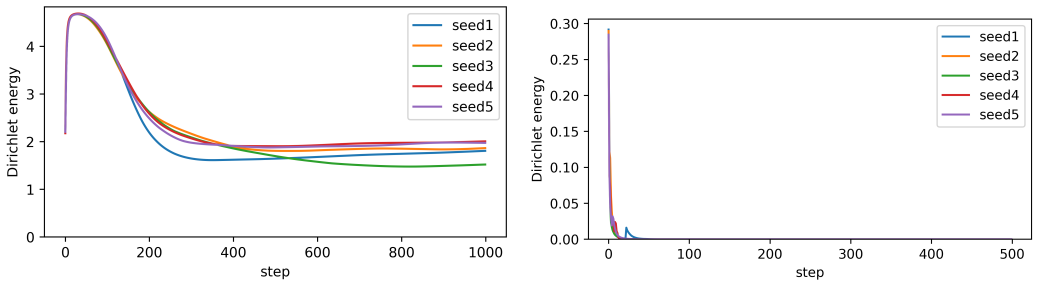
$$x_i(t+1) = \sigma(\boldsymbol{W}^{ne} \sum_{j \in \mathcal{N}(i)} e_{ij} + \boldsymbol{W}^n_\beta x_i(t)) \tag{12}$$
$$e_{ij}(t) = \sigma(\boldsymbol{W}^{en}(x_i(t) - x_j(t))) \quad \text{or}$$
$$e_{ij}(t) = \boldsymbol{W}^{en}(x_i(t) - x_j(t)),$$

for the same randomly initialized weights drawn from a normal distribution with spread 0.1. As discussed in the



(a) Dirac Bianconi 1-Step (DB1S) (b) Dirac Bianconi T-Step (c) DBGNN layer, where Lin denotes a linear layer.

Figure 2. Illustration of the Dirac–Bianconi Graph Neural Network (DBGNN) layer (c) and its components. The Dirac–Bianconi T-step layer (b) consists of multiple DB1S (a). By stacking multiple DB1S and applying them sequentially, information can be propagated along the graph.



(a) DBGNN with 1 layer and 1,000 steps per layer (b) GCN with 100 layers

Figure 3. Evolution of the normalized Dirichlet energy (DE) of the node feature embeddings for a sample of dataset20 with five different seeds and no training. For DBGNNs, the DE remains at a high level, meaning that information can be deeply propagated, while for GCNs, information is quickly lost.

introduction, Equation (10) can induce both oscillatory and non-oscillatory behavior, depending on the weights. Here we investigate the properties of the oscillatory regime, in which we might expect propagating waves. To do so, we constrain the weights to be $\boldsymbol{W}^{ne} = -\boldsymbol{W}^{en\dagger}$ and $\boldsymbol{W}_{\beta}^{n/e}$ antisymmetric.

We evolve these models on a 5x20 rectangular grid, where one of the short edges has all nodes randomly initialized, and all other edge and node features are identically zero. Figure 4 shows example trajectories for five models with four edge and node features to illustrate the behavior of DBGNNs in comparison to MPNNs. The linear DB equation shows a leading edge, a concentrated wave of activation that spreads rapidly into the network before dissipating, with ripples radiating into the rest of the network. Due to the oscillatory initialization, the linear MPNN equation also shows oscillatory behavior, but this does not result in spreading into the network. Adding the non-linearities stabilizes the leading edge of the DB equation, which now reaches the other end of the graph, and also sharpens the ripples into a coherent excitation that travels slower down the graph. For the MPNN, the non-linearities suppress the oscillations, leaving us with pure diffusion. For higher dimensional internal spaces, as well as for many non-oscillatory random weights, most configurations of all layers exhibit slow diffusion in the system. Occasionally, we can randomly generate coherent traveling excitations in the DBGNN, which are not observed in the MPNN. We provide examples of these trajectories in the appendix. We conclude that the wave aspects of the DBGNN enable deep propagation of signals into the graph, while edge non-linearities play a minor role.

6. Experimental results

We evaluate the performance of our DBGNN on challenging tasks involving long-range dependencies. The datasets deal with power grid properties and properties associated with the molecular structures of peptides. Power grids are known to encompass long-range dependencies, as highlighted by Ringsquandl et al. (2021). For the power grid dataset, GNNs capable of effectively propagating information over extended distances demonstrate superior performance (Nauck et al., 2023). The emphasis for this dataset is on topological relationships, given the absence of edge features and the presence of only one binary node

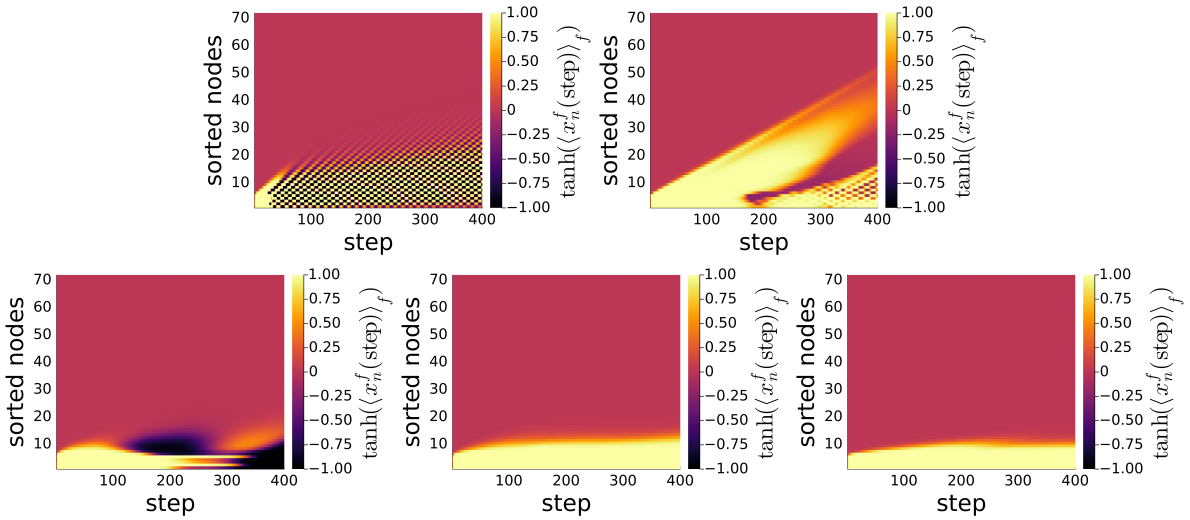


Figure 4. Oscillatory regime: Feature activation versus steps of the linear DB Equation (10) (top left), the non-linear DB 1-step layer Equation (10) + ReLU (top right), the linear MPNN layer Equation (11) (bottom left), and MPNN Equation (12) without (middle) and with non-linear messages (bottom right). $d_n = d_f = 4$, same random weights. In the case of DBGNN, information can be propagated and is not lost by diffusion.

feature. In contrast, the molecular dataset exhibits a diverse array of node and edge features. These peptide datasets were released with the specific purpose of benchmarking the handling of long-range dependencies by various GNNs. The selected tasks encompass node regression, graph regression, and graph classification.

Dynamic stability of power grids The most sophisticated dataset dealing with the dynamic stability of power grids is published by Nauck et al. (2023), which is based on Nauck et al. (2022b;a). There are a total of 20,000 grids: 10,000 small grids of size 20 (dataset20), and 10,000 medium-sized grids of size 100 (dataset100). Besides training and evaluating the models on the same grid sizes, we also analyze the out-of-distribution generalization by training the models on grids of size 20 and evaluating them on grids of size 100. We follow the nomenclature from Nauck et al. (2023) and refer to this task as tr20ev100.

Training models on smaller grids and evaluating them on larger grids is important for real-world applications because the computational effort increases at least quadratically with the size of the grid.

The results are given in Table 1, where we compare them with the current benchmark performances. The DBGNN achieves the best performance on all tasks and significantly outperforms the other models on out-of-distribution generalization. One of the reasons for its superior performance may be related to its ability to go deep without encountering the problem of oversmoothing. The final DBGNN consists of 4 DB 12-step layers, resulting in a total of 48 steps.

To further investigate the absence of smoothing, we compute the Dirichlet energy for a sample of dataset20 in the forward pass using the node embeddings at each step. Figure 5 shows that for trained models, the Dirichlet energy remains high throughout the forward pass, confirming the intuition that DBGNNs do not suffer from oversmoothing even at considerable depth. Some seeds go through periods of considerable "sharpening", especially following the change in dynamics after each T-step.

Table 1. Performance of dynamic stability prediction measured by the R^2 score in % using the benchmark models from Nauck et al. (2023), distinguishing between in- and out-of-distribution (distr.) The columns *tr20ev20* and *tr100ev100* indicate that the models are trained and evaluated on the same datasets. The out-of-distribution performance is measured by evaluating the models on dataset100 after training them on dataset20 (tr20ev100). Following Nauck et al. (2023), we show the mean performance of the best three configurations out of five different initializations.

Model	**In-distr.**		**Out-of-distr.**
	tr20ev20	tr100ev100	tr20ev100
ArmaNet	82.22 $\pm$ 0.12	88.35 $\pm$ 0.12	67.12 $\pm$ 0.80
GCNNet	70.74 $\pm$ 0.15	75.19 $\pm$ 0.14	58.24 $\pm$ 0.47
TAGNet	82.50 $\pm$ 0.36	88.32 $\pm$ 0.10	66.32 $\pm$ 0.74
DBGNN	**85.68** $\pm$ 0.10	**90.08** $\pm$ 0.02	**73.73** $\pm$ 0.07

Peptide property prediction To assess the performance of the DBGNN model on long-range interactions, we use the *Peptides-struct* dataset from the long-range benchmark dataset (Dwivedi et al., 2023). Peptides, defined as short chains of amino acids, play a crucial role in numerous biological processes. Given the intricate relationships between

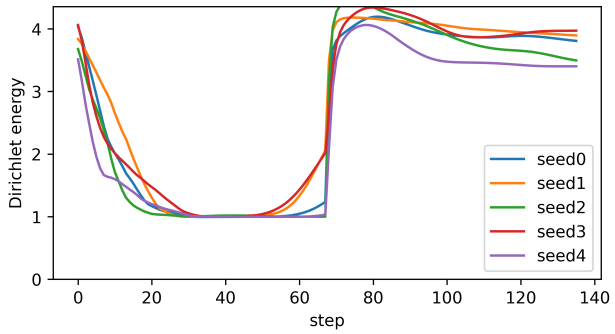


Figure 5. Evolution of normalized Dirichlet energy of node feature embeddings in a trained DBGNN layer for a sample of dataset20 with five different seeds.

Table 2. Performance on *Peptides-struct* visualized by mean absolute error (MAE). The performance of the baseline is taken from Dwivedi et al. (2023). The number of parameters is shown in column # param.

Model	**# param**	train MAE	test MAE
GCN	508k	0.2939 ±0.0055	0.3496 ±0.0013
GCNII	505k	0.2957 ±0.0025	0.3471 ±0.0010
GINE	476k	0.3116 ±0.0047	0.3547 ±0.0045
GatedGCN	509k	0.2761±0.0032	0.3420 ±0.0013
GatedGCN+RWSE	506k	0.2578 ±0.0116	0.3357 ±0.0006
Transform+LapPE	488k	0.2403 ±0.0066	**0.2529** ±0.0016
SAN+LapPE	493k	0.2822 ±0.0108	0.2683 ±0.0043
SAN+RWSE	500k	0.2680 ±0.0038	0.2545 ±0.0012
DBGNN	**127k**	**0.2433** ±0.0044	0.2864 ±0.0046

peptides and their biological functions, computational prediction of peptide properties is crucial for advancing drug development and biomolecular engineering. The peptide datasets contain 15,535 graphs, with an average of 150.94 nodes per graph and an average diameter of 56.99.

The *Peptides-struct* dataset is a multi-label graph regression task. Various properties of the peptides are predicted, with the full details presented in Dwivedi et al. (2023).

The performance of the baselines from Dwivedi et al. (2023) and the DBGNN is shown in Table 2. The baselines make use of additional methods such as random-walk structural encoding (RWSE) (Dwivedi et al., 2022) and Laplacian positional encoding (LapPE) (Dwivedi et al., 2020) to improve performance. Spectral attention networks (SANs) are improved transformers introduced by Kreuzer et al. (2021). Tönshoff et al. (2023) show that the performance of MPNNs such as GCNs can be significantly improved by adding multi-layer regression heads to the output of GNNs. However, we refrain from including multi-head regressions in order to focus on the pure properties of message passing components. The DBGNN, consisting of only about a quarter of the parameters, still significantly outperforms all message passing graph neural networks, but achieves lower performance than the transformers. We conclude that the DBGNN is a powerful MPNN capable of dealing with node and edge features and correctly identifying long-range dependencies.

Comparison with other oversmoothing techniques Considering the inherent flaw of oversmoothing in GNNs and especially in MPNNs, various methods have been introduced to mitigate the issue (Rusch et al., 2023). These approaches include diverse strategies such as applying normalization techniques, employing regularization methods, incorporating residual connections, and modifying the dynamics of GNNs (Rusch et al., 2023). Residual connections, exemplified by models such as GatedGCN (Bresson & Laurent, 2018) and GCNII (Chen et al., 2020), provide a way to address oversmoothing. GCNII is specifically designed to tackle oversmoothing using **I**nitial residual and **I**dentity mapping. Distinct but related methods include ArmaNet (Bianchi et al., 2021) and TAGNet (Du et al., 2017), which involve multiple steps within a single layer, enabling the construction of deeper GNNs that provide meaningful output. To address oversmoothing in MPNNs, Kreuzer et al. (2021) propose SANs and use positional encoding techniques.

In contrast, the DBGNN motivated by dynamical systems has several interesting implications. One is that it is inherently non-oversmoothing, as demonstrated in Section 5. This intrinsic attribute potentially contributes to the outstanding performance of the DBGNN across various datasets, as demonstrated in Tables 1 and 2, which may provide valuable insights for the development of enhanced GNN models.

7. Conclusion

We introduced a new graph neural network layer based on a straightforward generalization of the topological Dirac–Bianconi equation on networks. We show that this model has no intrinsic tendency to equilibrate features on the network. This has the potential to handle long-range dependencies and treat edge and node features equally. By incorporating multiple steps with weight sharing within a layer, we enable the layer to efficiently learn dynamics that probe the graph deeply. The DBGNN is a straightforward adaptation of the topological Dirac–Bianconi equation, and thus offers the potential for many further modifications. In its current form, the DBGNN already outperforms other layers in predicting the dynamic stability of power grids and achieves competitive performance in predicting molecular properties, outperforming conventional MPNN methods.

By analyzing the internal node embeddings using the Dirichlet energy, we can show that the DBGNN does not seem to suffer from the oversmoothing problem. Furthermore, we observe a sudden sharpening of the features when the dynamics change after many steps, a phenomenon that needs to be better understood. We provide evidence that

the long-range capabilities result from the underlying Dirac–Bianconi dynamics rather than from edge non-linearity. The experiments were performed without adapting the model to the task at hand. Since the DBGNN is close to standard MPNN-style networks, the vast array of modifications that exist for them can be easily adapted here. It remains to be seen whether they can also enhance DBGNN-style networks. One particularly interesting question is whether long-range modifications to GCNs, such as Gutteridge et al. (2023), can further enhance the long-range behavior of DBGNNs.

Overall, the expanded long-range capabilities provide new opportunities for power grid analysis and molecular prediction, offering increased potential for scientific exploration and understanding.

8. Acknowledgements

All authors gratefully acknowledge Land Brandenburg for supporting this project by providing resources on the high-performance computer system at the Potsdam Institute for Climate Impact Research. The work was in part supported by DFG Grant Numbers KU 837/39-2 (360460668), BMWK Grant 03EI1016A, and BMBF Grant 03SF0766. Christian Nauck would like to thank the German Federal Environmental Foundation (DBU) for funding his PhD scholarship and Professor Jörg Raisch for supervising his PhD. Michael Lindner greatly acknowledges support from the Berlin International Graduate School in Model and Simulation (BIMoS) and from his doctoral supervisor Professor Eckehard Schöll. Rohan Gorantla was supported by the United Kingdom Research and Innovation (grant EP/S02431X/1), UKRI Centre for Doctoral Training in Biomedical AI at the University of Edinburgh, School of Informatics and Exscientia Plc, Oxford. AI tools are used on the (sub-)sentence level to improve language. We especially want to thank the reviewers for their valuable comments and feedback, which have significantly enhanced the quality of this paper. We also express our gratitude to Teresa Gehrs from LinguaConnect and the Proofreading Service for Junior Scholars at TU Berlin for their help in proofreading this paper.

References

Acebrón, J. A., Bonilla, L. L., Pérez Vicente, C. J., Ritort, F., and Spigler, R. The Kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of Modern Physics*, 77(1):137–185, April 2005. doi: 10.1103/RevModPhys.77.137.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Besard, T., Foket, C., and De Sutter, B. Effective Extensible Programming: Unleashing Julia on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 30(4): 827–841, April 2019. ISSN 1558-2183. doi: 10.1109/TPDS.2018.2872064.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM Review*, 2017. ISSN 00361445. doi: 10.1137/141000671.

Bianchi, F. M., Grattarola, D., Livi, L., and Alippi, C. Graph Neural Networks with Convolutional ARMA Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (01):1–1, January 2021. ISSN 0162-8828. doi: 10.1109/TPAMI.2021.3054830.

Bianconi, G. The topological Dirac equation of networks and simplicial complexes. *Journal of Physics: Complexity*, 2(3):035022, September 2021. ISSN 2632-072X. doi: 10.1088/2632-072X/ac19be.

Blakely, D., Lanchantin, J., and Qi, Y. Time and Space Complexity of Graph Convolutional Networks. 2021. URL https://qdata.github.io/deep2Read/talks-mb2019/Derrick_201906_GCN_complexityAnalysis-writeup.pdf.

Bodnar, C., Frasca, F., Wang, Y. G., Otter, N., Montúfar, G., Liò, P., and Bronstein, M. Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks, June 2021.

Bodnar, C., Di Giovanni, F., Chamberlain, B., Lió, P., and Bronstein, M. Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs. *Advances in Neural Information Processing Systems*, 35: 18527–18541, December 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/75c45fca2aa416ada062b26cc4fb7641-Abstract-Conference.html.

Bresson, X. and Laurent, T. Residual Gated Graph ConvNets, April 2018.

Bunch, E., You, Q., Fung, G., and Singh, V. Simplicial 2-Complex Convolutional Neural Nets, December 2020.

Chamberlain, B., Rowbottom, J., Eynard, D., Di Giovanni, F., Dong, X., and Bronstein, M. Beltrami Flow and Neural Diffusion on Graphs. In *Advances in Neural Information Processing Systems*, volume 34, pp. 1594–1609. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/0cbed40c0d920b94126eaf5e707be1f5-Abstract.html.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. GRAND: Graph Neural Diffusion. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 1407–1418. PMLR, July 2021b. URL https://proceedings.mlr.press/v139/chamberlain21a.html.

Chen, J. and Chen, H. Edge-Featured Graph Attention Network, January 2021. URL http://arxiv.org/abs/2101.07671.

Chen, J., Wang, Y., Bodnar, C., Ying, R., Lio, P., and Wang, Y. G. Dirichlet Energy Enhancement of Graph Neural Networks by Framelet Augmentation, November 2023.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and Deep Graph Convolutional Networks, July 2020.

Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., and Bronstein, M. M. Understanding convolution on graphs via energies, September 2023. URL http://arxiv.org/abs/2206.10991.

Du, J., Zhang, S., Wu, G., Moura, J. M. F., and Kar, S. Topology Adaptive Graph Convolutional Networks. *arxiv*, October 2017. doi: 10.48550/arXiv.1710.10370.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking Graph Neural Networks. *arXiv:2003.00982 [cs, stat]*, July 2020. URL http://arxiv.org/abs/2003.00982.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph Neural Networks with Learnable Structural and Positional Representations, February 2022.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long Range Graph Benchmark, January 2023.

Eliasof, M., Haber, E., and Treister, E. PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations, October 2021.

Eliasof, M., Haber, E., and Treister, E. Feature Transportation Improves Graph Neural Networks, December 2023.

Fey, M. and Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric. arxiv preprint, 2019. doi: 10.48550/ARXIV.1903.02428.

Fu, G., Dupty, M. H., Dong, Y., and Sun, L. W. Implicit Graph Neural Diffusion Based on Constrained Dirichlet Energy Minimization, August 2023. URL http://arxiv.org/abs/2308.03306.

Gutteridge, B., Dong, X., Bronstein, M., and Di Giovanni, F. DRew: Dynamically Rewired Message Passing with Delay, May 2023.

He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. A Generalization of ViT/MLP-Mixer to Graphs, May 2023.

Innes, M., Saba, E., Fischer, K., Gandhi, D., Rudilosso, M. C., Joy, N. M., Karmali, T., Pal, A., and Shah, V. Fashionable Modelling with Flux, November 2018.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*, February 2017. doi: arXiv:1609.02907.

Kreuzer, D., Beaini, D., Hamilton, W. L., Létourneau, V., and Tossou, P. Rethinking Graph Transformers with Spectral Attention, October 2021.

Kuramoto, Y. Self-entrainment of a population of coupled non-linear oscillators. In Araki, H. (ed.), *International Symposium on Mathematical Problems in Theoretical Physics 1975*, pp. 420–422, Berlin, Heidelberg, 1975. Springer. ISBN 978-3-540-37509-8. doi: 10.1007/BFb0013365.

Liu, X., Zhou, B., Zhang, C., and Wang, Y. G. Framelet Message Passing, February 2023.

Lloyd, S., Garnerone, S., and Zanardi, P. Quantum algorithms for topological and geometric analysis of data. *Nature Communications*, 7(1):10138, January 2016. ISSN 2041-1723. doi: 10.1038/ncomms10138.

Lucibello, C. GraphNeuralNetworks.jl, July 2023. URL https://github.com/CarloLucibello/GraphNeuralNetworks.jl.

Maskey, S., Paolino, R., Bacho, A., and Kutyniok, G. A Fractional Graph Laplacian Approach to Oversmoothing, October 2023.

Menck, P. J., Heitzig, J., Marwan, N., and Kurths, J. How basin stability complements the linear-stability paradigm. *Nature Physics*, 9(2):89–92, February 2013. ISSN 1745-2481. doi: 10.1038/nphys2516.

Nauck, C., Lindner, M., Schürholt, K., and Hellmann, F. Towards dynamic stability analysis of sustainable power grids using graph neural networks. NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning, December 2022a. doi: 10.48550/arXiv.2212.11130.

Nauck, C., Lindner, M., Schürholt, K., Zhang, H., Schultz, P., Kurths, J., Isenhardt, I., and Hellmann, F. Predicting basin stability of power grids using graph neural networks. *New Journal of Physics*, 24(4):043041, April 2022b. ISSN 1367-2630. doi: 10.1088/1367-2630/ac54c9.

Nauck, C., Lindner, M., Schürholt, K., and Hellmann, F. Toward dynamic stability assessment of power grid topologies using graph neural networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(10):103103, October 2023. ISSN 1054-1500. doi: 10.1063/5.0160915.

Nitzbon, J., Schultz, P., Heitzig, J., Kurths, J., and Hellmann, F. Deciphering the imprint of topology on nonlinear dynamical network stability. *New Journal of Physics*, 19 (3):033029, March 2017. ISSN 1367-2630. doi: 10.1088/1367-2630/aa6321.

Oono, K. and Suzuki, T. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification, January 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. Graph Neural Ordinary Differential Equations, June 2021.

Ringsquandl, M., Sellami, H., Hildebrandt, M., Beyer, D., Henselmeyer, S., Weber, S., and Joblin, M. Power to the Relational Inductive Bias: Graph Neural Networks in Electrical Power Grids. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 1538–1547, New York, NY, USA, October 2021. Association for Computing Machinery. ISBN 978-1-4503-8446-9. doi: 10.1145/3459637.3482464.

Rodrigues, F. A., Peron, T. K. D., Ji, P., and Kurths, J. The Kuramoto model in complex networks. *Physics Reports*, 610:1–98, January 2016. ISSN 0370-1573. doi: 10.1016/j.physrep.2015.10.008.

Rossi, E., Charpentier, B., Di Giovanni, F., Frasca, F., Günnemann, S., and Bronstein, M. Edge Directionality Improves Learning on Heterophilic Graphs, November 2023.

Rusch, T. K., Chamberlain, B. P., Rowbottom, J., Mishra, S., and Bronstein, M. M. Graph-Coupled Oscillator Networks, June 2022.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A Survey on Oversmoothing in Graph Neural Networks, March 2023.

Tönshoff, J., Ritzert, M., Rosenbluth, E., and Grohe, M. Where Did the Gap Go? Reassessing the Long-Range Graph Benchmark, September 2023.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature, November 2022.

Wang, Y., Yi, K., Liu, X., Wang, Y. G., and Jin, S. ACMP: Allen-Cahn Message Passing for Graph Neural Networks with Particle Phase Transition. 2023. doi: 10.48550/ARXIV.2206.05437.

Witthaut, D., Hellmann, F., Kurths, J., Kettemann, S., Meyer-Ortmanns, H., and Timme, M. Collective nonlinear dynamics and self-organization in decentralized power grids. *Reviews of Modern Physics*, 94(1):015005, February 2022. doi: 10.1103/RevModPhys.94.015005.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-2388. doi: 10.1109/TNNLS.2020.2978386.

Yang, Y. and Li, D. NENN: Incorporate Node and Edge Features in Graph Neural Networks. In *Proceedings of The 12th Asian Conference on Machine Learning*, pp. 593–608. PMLR, September 2020. URL https://proceedings.mlr.press/v129/yang20a.html.

Zhao, K., Kang, Q., Song, Y., She, R., Wang, S., and Tay, W. P. Graph Neural Convection-Diffusion with Heterophily, May 2023. URL http://arxiv.org/abs/2305.16780.

Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S.-H., and Hu, X. Dirichlet Energy Constrained Learning for Deep Graph Neural Networks, July 2021.

Zhou, Y., Huo, H., Hou, Z., Bu, L., Mao, J., Wang, Y., Lv, X., and Bu, F. Co-embedding of edges and nodes with deep graph convolutional neural networks. *Scientific Reports*, 13(1):16966, October 2023. ISSN 2045-2322. doi: 10.1038/s41598-023-44224-1.

A. Appendix

A.1. Data availability

The code to train the DBGNNs and to generate the figures is available on https://github.com/PIK-ICoNe/DBGNN_paper-companion.git and https://doi.org/10.5281/zenodo.12687981.

A.2. Dirichlet energy

The Dirichlet energy is a measure of the heterogeneity of features across the graph. The normalized Dirichlet energy (DE) is computed by:

$$\text{Dirichlet energy} = \frac{\text{tr}(x(k)^\top \boldsymbol{L} x(k))}{\text{tr}(x(k)^\top x(k))}, \tag{13}$$

where $x(k)$ denotes the node embedding after k steps and tr denotes the trace operator. The interpretation becomes apparent by rewriting the Dirichlet energy in terms of edge differences:

$$\text{Dirichlet energy} = \frac{\sum_{(i,j)\in\mathcal{E}} ||x^i(k) - x^j(k)||^2}{\sum_{i\in\mathcal{N}} ||x^i(k)||^2}, \tag{14}$$

where $x^i(k) \in F_n$ denotes the state of node i after k steps.

A.3. Hyperparameters for reproducibility

The following tables provide the hyperparameters to reproduce the main results. For the power grid dataset, the information is given in Table 3, and for the peptide structure task in Table 4.

Table 3. Properties of DBGNN after hyperparameter studies for power grid datasets

parameter	dataset with grids of size 20	dataset with grids of size 100
number of layers (K)	2	2
number of steps (with shared weights) per layer (T)	68	68
d_n^{hidden}	113	113
d_e^{hidden}	109	109
dropout for node convolution	$\approx 1.4 \times 10^{-2}$	$\approx 2.1 \times 10^{-2}$
dropout for edge convolution	$\approx 1.9 \times 10^{-3}$	5.7×10^{-4}
batch size	50	50
epochs	2 000	2 000
learning rate (LR)	max 6.1×10^{-4}	max: $\approx 2.6 \times 10^{-3}$
scheduler	oneCycleLR with initial div factor: 32, final div factor: 5.8×10^5	oneCycleLR with initial div factor: 75, final div factor: $\approx 1.7 \times 10^6$

A.4. Implementation and computation details

The packages *GraphNeuralNetworks.jl*(Lucibello, 2023) and *Flux.jl* (Innes et al., 2018) are used for the Julia implementation of the DBGNN (Bezanson et al., 2017). Furthermore, *Cuda.jl* (Besard et al., 2019) and *MLDatasets.jl* are used. We also provide a PyTorch (Paszke et al., 2019) implementation using PyTorch Geometric (Fey & Lenssen, 2019). We run all experiments on NVIDIA V100 and H100 accelerators.

For the power grid task and tr20ev20, training five seeds in parallel on one V100 takes about three days and 20 hours for 2,000 epochs, but significant overfitting occurs after roughly 500 epochs. For tr100ev100, overfitting occurs after roughly 600 epochs, and training two seeds in parallel take four days and eight hours for 2,000 epochs. For peptides-struct, 2,000 epochs with two seeds in parallel take roughly five days and 13 hours. For peptides-func, training two seeds in parallel takes about one day and 18 hours in total for 500 epochs.

Table 4. Properties of the DBGNN after hyperparameter studies for peptide structure datasets

parameter	*peptides-struct*
number of layers (K)	2
number of steps (with shared weights) per layer (T)	68
d_n^{hidden}	113
d_e^{hidden}	109
dropout for node convolution	≈ 0.0554
dropout for edge convolution	≈ 0.01058
batch size	60
epochs	2000
max. learning rate (LR)	≈ 0.00548
oneCyle scheduler	init div factor: 151, final div factor: $\approx 4.217 \times 10^9$

A.5. Computational complexity of the DBGNN

The computational complexity is mostly determined by the repeated application of LinDB from Equation (10), which we will focus on in the following. Hence, we neglect the mapping of the input features to the hidden feature spaces, which only needs to be done once before applying the DBGNN steps, as well as the concatenation of matrices and the application of non-linearities. For the expensive LinDB layer, we make use of sparse operators, such as PyTorch Scatter (Paszke et al., 2019), by exploiting the sparsity of the adjacency matrix to reduce the computational effort, which is explained in more detail in (Blakely et al., 2021). The following paragraph describes the computational complexity in more detail.

For simplicity, we assume that $d_e = d_n$, meaning that the hidden dimension of node and edge features is the same. There are two relevant terms of the sum for node and edge propagation, respectively. First, for node propagation, using the coupling matrix $\boldsymbol{W}^{ne}$ and multiplying it by the edge features $F_e^{\mathcal{E}}$ for all neighbors yields: $\mathcal{O}(|E| \times d_n)$. The second term, which is a dense matrix multiplication of W_β^n and the node features $F_n^{\mathcal{N}}$ yields: $\mathcal{O}(d_n \times d_n \times |E|)$. Similarly, for the propagation of edges, we have $\mathcal{O}(|N| \times d_n)$ and the multiplication of $\boldsymbol{W}_\beta^e \times F_e^{\mathcal{E}}$ yields: $\mathcal{O}(d_e \times d_e \times |E|)$. These multiplications are repeated T-times, where T denotes the number of DB steps. Hence, we get: $\mathcal{O}\left(T(|E| \times d_n + |N| \times d_n + d_n^2 \times |E| + d_n^2 \times |N|)\right)$, which is comparable to the effort of the GCN (Blakely et al., 2021).

The runtime of the implemented layer can be improved by utilizing the sparsity of the adjacency matrix for the matrix multiplication as well, which does not work in the current implementation.

A.6. Experimental details on power grids

The dynamics of power grids is characterized by complex collective phenomena that extend over the whole system (Witthaut et al., 2022). The chosen task is based on the so-called single-node basin stability, originally introduced by (Menck et al., 2013), which describes nodal dynamic stability. It is the result of expensive Monte Carlo simulations and quantifies the probabilistic behavior of the entire power grid after applying nodal perturbations.

For the power grid models, the nodal input features are categorical representations of sources or sinks. The power lines are assumed to be homogeneous; therefore, the input edge features are simply set to 1. The absence of diverse features puts the emphasis on topological properties.

The datasets contain individual training, validation, and test sets (70:15:15). The only input feature per node, which describes whether a node is considered a source ($P = 1$) or sink ($P = -1$), is based on the injected power P. Since homogeneous coupling is used, there are no edge features. Performance on the nodal regression setup is evaluated using the coefficient of determination (R^2).

A.7. Non-oscillatory random weights

Figure 6 provides example trajectories with random initial weights for which the MPNN and the DBGNN do not differ significantly. We suspect that random initialization leads to a washout of the directionality required to generate coherent propagation. How to robustly generate traveling activation remains an open research question.

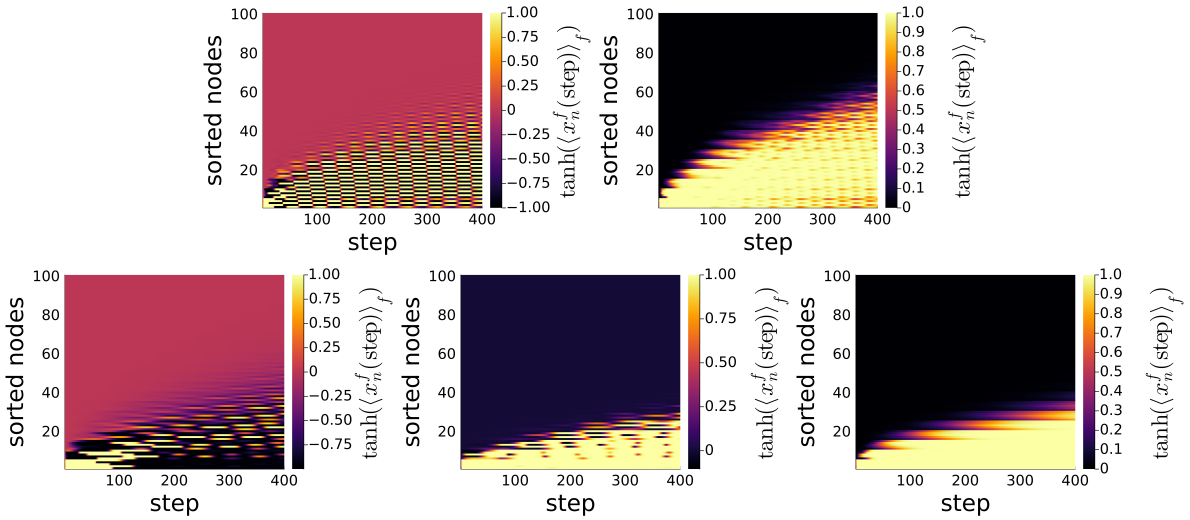


Figure 6. Non-oscillatory regime: Feature activation versus steps of the linear DB Equation (10) (top left), the non-linear DB 1-step layer Equation (10) + ReLU (top right), the linear MPNN layer Equation (11) (bottom left), and the MPNN Equation (12) without (middle) and with non-linear messages (bottom right). $d_n = d_f = 4$, same random weights.

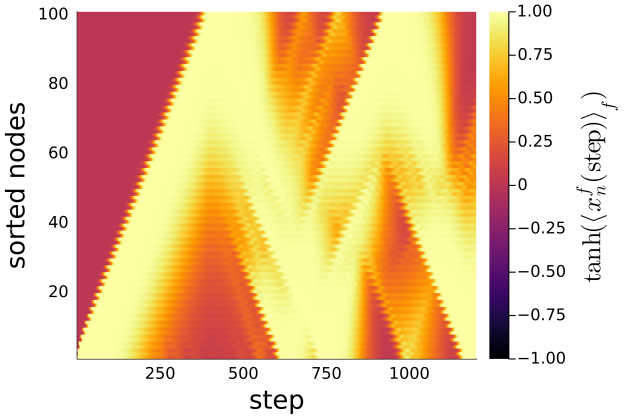


Figure 7. Oscillatory regime, full DB 1-step layer Equation (10) + non-linearity, ladder graph, $d_n = d_f = 4$.

We also provide two videos in the supplementary materials DB that show how a DBGNN layer can have activation that travels along a ladder graph and is reflected at the far edge compared to MPNN. The trajectory is provided in Figure 7.

B. Limitations

The introduced DBGNN uses the Dirac operator, but no other special methods that could improve performance. It would be interesting to integrate other techniques, such as attention, to obtain better performance.

C. Social impact

We present a potentially powerful GNN for analyzing graph-structured data. We currently have no reason to believe that the new method introduces specific negative societal impacts beyond the general drawbacks of ML in general and improved graph neural networks in particular.