Mila
UNIVERSITY OF CAMBRIDGE
Valence

DIRECTIONAL GRAPH NETWORKS (DGN)

Dominique Beaini*, Saro Passaro*, Vincent Létourneau, William Hamilton, Gabriele Corso, Pietro Liò
dominique@valencediscovery.com, sp976@cam.ac.uk

arxiv.org/abs/2010.02863

@dom_beaini
@saro2000
@invivo_ai

Problematic – GNNs lack expressiveness

Properties	GNN	DGN
Aggregate neighbouring features	✓	✓
Anisotropy based on node features/attention	✓	✓
Globally consistent directional flow		✓
Efficiently send messages across the longest length of the graph		✓
Avoids over-smoothing and over-squashing		✓
More powerful than the 1-WL isomorphism test		✓
Can replicate Gabor-like filters		✓
Generalize CNNs		✓

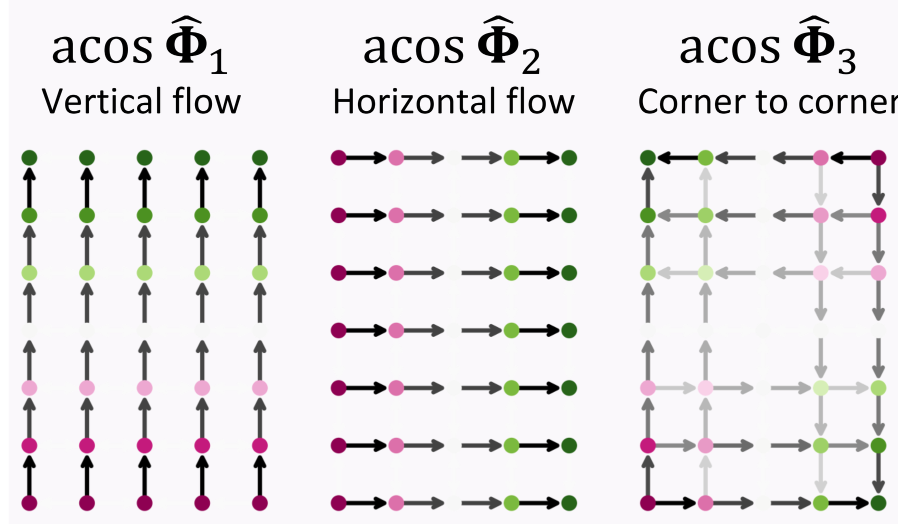
GNN: Graph neural network
CNN: Convolutional neural network
ϕ_k: k-th non-zero lowest frequency eigenvector
WL : Weisfeiler-Lehman

Low frequency eigenvectors ϕ_k

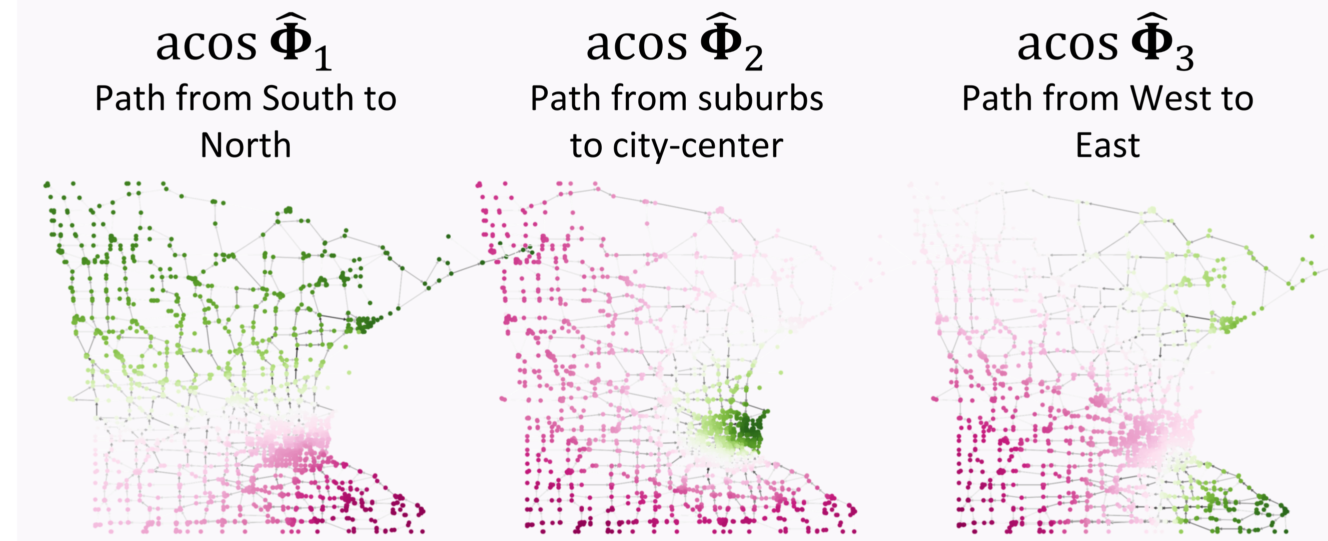
Low frequency eigenvectors are a representation of the global structure of the graph and are robust to small changes.

The gradient of the eigenvectors flow in interpretable directions.

Grid graph (7 × 5)



Minnesota highway roadmap



Empirical results

Model	ZINC No edge features MAE	ZINC Edge features MAE	PATTERN No edge features % acc	CIFAR10 No edge features % acc	CIFAR10 Edge features % acc	MolHIV No edge features % ROC-AUC
GCN	0.469		65.88	54.46		76.06
GIN	0.408		85.59	53.38		75.58
GraphSage	0.41		50.52	66.08		
GAT	0.463		75.82	65.48		
MoNet	0.407		85.48	53.43		
GatedGCN	0.422	0.363	84.48	69.19	69.37	
PNA	0.32	0.188	86.57	70.46	70.47	79.05
DGN (ours)	**0.219**	**0.168**	**86.68**	**72.70**	**72.84**	**79.70**

Pre-computed steps $O(kE)$

Input graph

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

Compute first k non-trivial eigenvectors

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

Compute the gradient

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.

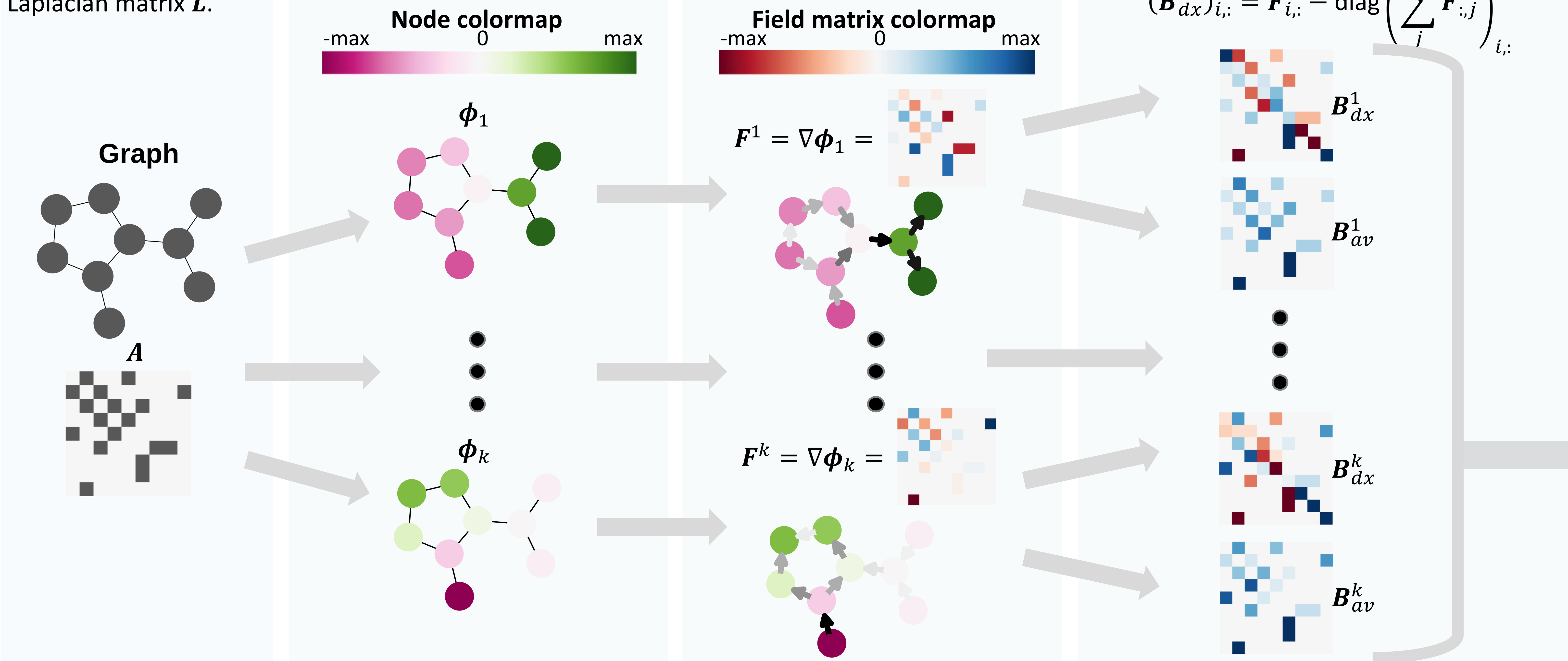
Create the aggregation matrices B

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

$$\boldsymbol{B}_{av} = |\hat{\boldsymbol{F}}|$$

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

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



Graph neural network steps $O(kE + kN)$

Input graph

The feature matrix $\boldsymbol{X}^{(0)}$ has N rows (the number of nodes) and n_0 columns (the number of input features).

Aggregation of neighbouring features

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

Other non-directional aggregators are used, such as the mean aggregation $\boldsymbol{D}^{-1}\boldsymbol{A}\,\boldsymbol{X}^{(0)}$.

MLP

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 $\boldsymbol{Y}$.