Graph Neural Networks and Diffusion PDEs

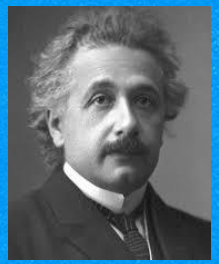
Ben Chamberlain, James Rowbottom

@b_p_chamberlain, @_JRowbottom

Why:

- We show a GNN can be seen as a discretisation of a PDE
- The study of PDEs has existed far longer than GNNs
- Many of histories greatest minds have helped develop tools to understand properties of PDEs



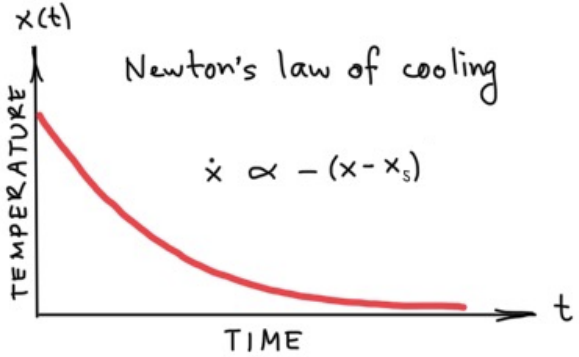



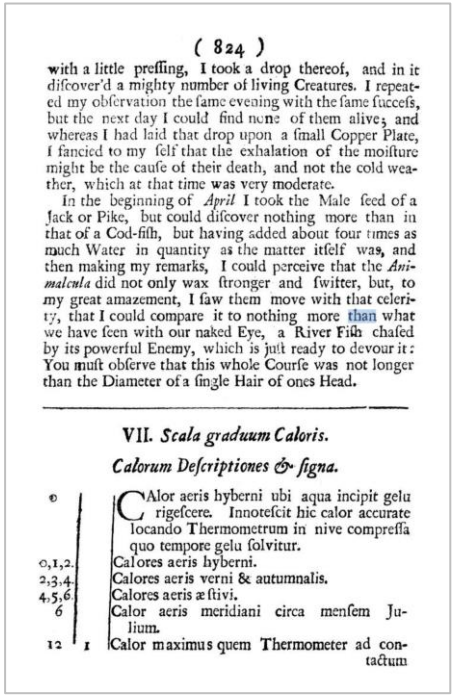

Diffusion PDEs
Connecting PDEs to GNNs
Diffusion in image processing

GRAND
Positional Encodings
BLEND

Diffusion PDEs

Newton Law of Cooling: "the temperature a hot body loses in a given time is proportional to the temperature difference between the object and the environment"




(824)

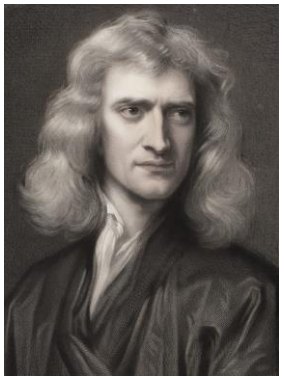
with a little preſſing, I took a drop thereof, and in it diſcover'd a mighty number of living Creatures. I repeated my obſervation the ſame evening with the ſame ſucceſs, but the next day I could find none of them alive; and whereas I had laid that drop upon a ſmall Copper Plate, I fancied to my ſelf that the exhalation of the moiſture might be the cauſe of their death, and not the cold weather, which at that time was very moderate.

In the beginning of *April* I took the Male ſeed of a Jack or Pike, but could diſcover nothing more than in that of a Cod-fiſh, but having added about four times as much Water in quantity as the matter itſelf was, and then making my remarks, I could perceive that the *Animalcula* did not only wax ſtronger and ſwifter, but, to my great amazement, I ſaw them move with that celerity, that I could compare it to nothing more than what we have ſeen with our naked Eye, a River Fiſh chaſed by its powerful Enemy, which is juſt ready to devour it: You muſt obſerve that this whole Courſe was not longer than the Diameter of a ſingle Hair of ones Head.

VII. *Scala graduum Caloris.*

Calorum Deſcriptiones & ſigna.

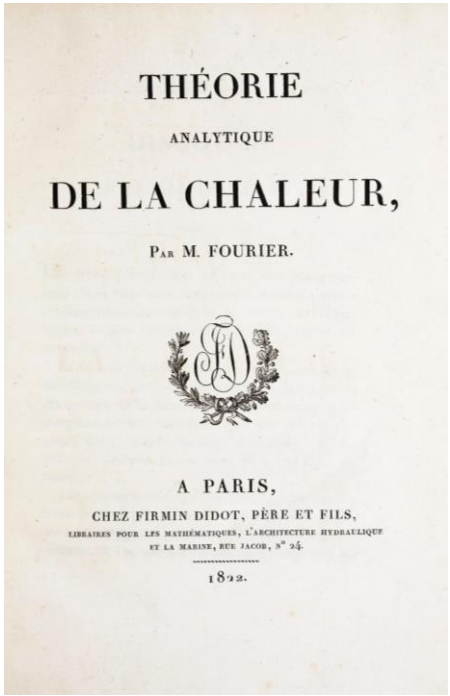
0		CAlor aeris hyberni ubi aqua incipit gelu rigeſcere. Innoteſcit hic calor accurate locando Thermometrum in nive compreſſa quo tempore gelu ſolvitur.
0,1,2.		Calores aeris hyberni.
2,3.4		Calores aeris verni & autumnalis.
4,5.6		Calores aeris æſtivi.
6		Calor aeris meridiani circa menſem Julium.
12	1	Calor maximus quem Thermometer ad contactum



Isaac Newton

Anonymous 1701

Fourier Heat Conduction Law:
"heat flux resulting from thermal conduction is proportional to the magnitude of the temperature gradient and opposite to it in sign"

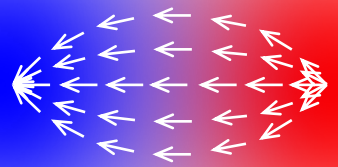

THÉORIE
ANALYTIQUE
DE LA CHALEUR,
PAR M. FOURIER.
A PARIS,
CHEZ FIRMIN DIDOT, PÈRE ET FILS,
LIBRAIRES POUR LES MATHÉMATIQUES, L'ARCHITECTURE HYDRAULIQUE ET LA MARINE, RUE JACOB, N° 24.
1822.

Joseph Fourier

Fourier 1822

Diffusion Equation

heat flux $h \propto -\nabla x$



Adolf Eugen Fick

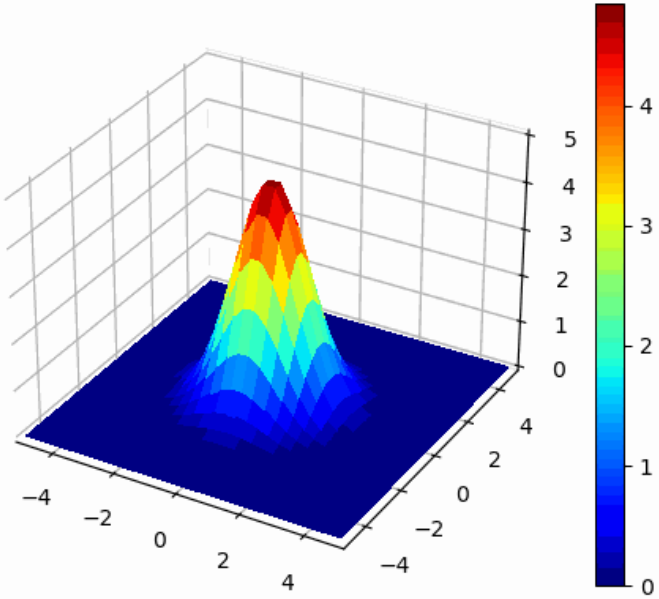
conservation condition: $\frac{\partial}{\partial t}x = -\mathrm{div}(h)$
("no heat created or disappears")

$$\frac{\partial}{\partial t}x = -\mathrm{div}(\nabla x)$$

Fick 1855

Diffusion takes many forms:

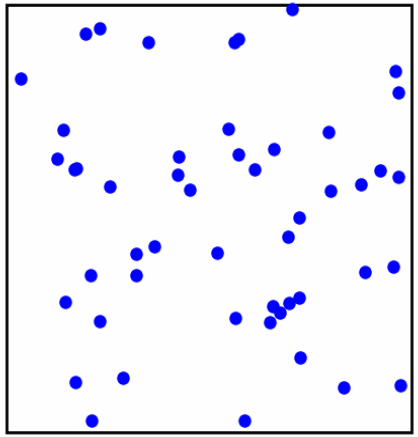
Heat diffusion:



Joseph Fourier

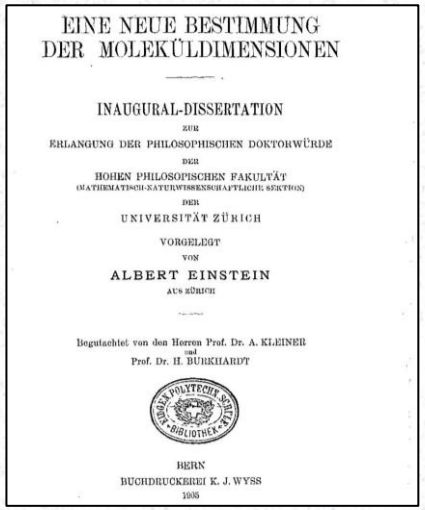
$$\frac{\partial x}{\partial t} = \Delta x$$

Diffusion takes many forms



Particle random walk: $dx_t = \mu dt + \sigma dW_t$

Particle density diffusion: $\frac{\partial \rho}{\partial t} = c\Delta\rho$

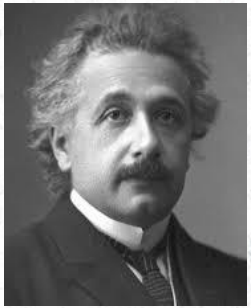


EINE NEUE BESTIMMUNG
DER MOLEKÜLDIMENSIONEN

INAUGURAL-DISSERTATION
ZUR
ERLANGUNG DER PHILOSOPHISCHEN DOKTORWÜRDE
DER
HOHEN PHILOSOPHISCHEN FAKULTÄT
(MATHEMATISCH-NATURWISSENSCHAFTLICHE SEKTION)
DER
UNIVERSITÄT ZÜRICH
VORGELEGT
VON
ALBERT EINSTEIN
AUS ZÜRICH

Begutachtet von den Herren Prof. Dr. A. KLEINER
und
Prof. Dr. H. BURKHARDT

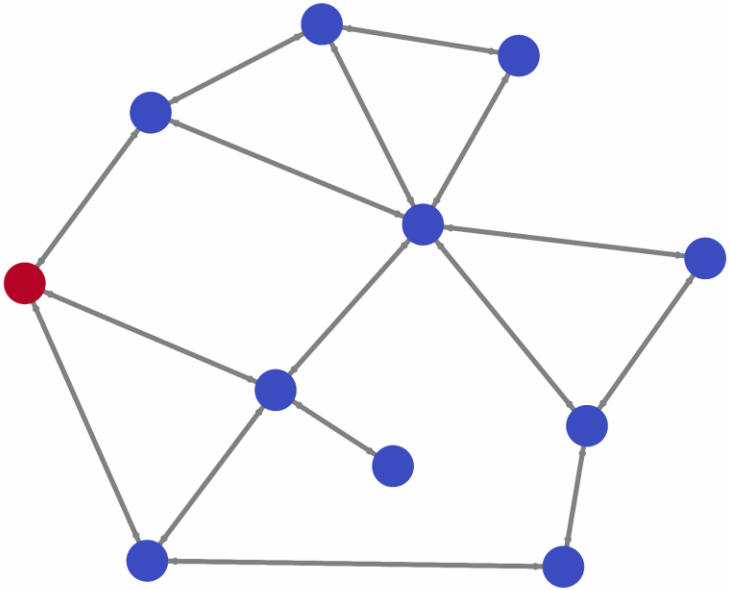
BERN
BUCHDRUCKEREI K. J. WYSS
1905



Albert Einstein

Diffusion takes many forms

Laplacian diffusion on graphs: $X_t = LX_{t-1}$



Pierre-Simon Laplace

Diffusion Equation

$$\frac{\partial}{\partial t}x = \Delta x$$

Homogeneous
Isotropic

Position-dependent diffusivity

$$\frac{\partial}{\partial t}x = -\mathrm{div}(a\nabla x)$$

Non-homogeneous
Isotropic

Position & direction dependent diffusivity

$$\frac{\partial}{\partial t}x = -\mathrm{div}(\mathbf{A}\nabla x)$$

Non-homogeneous
Anisotropic

Fick 1855

Connecting PDEs to GNNs

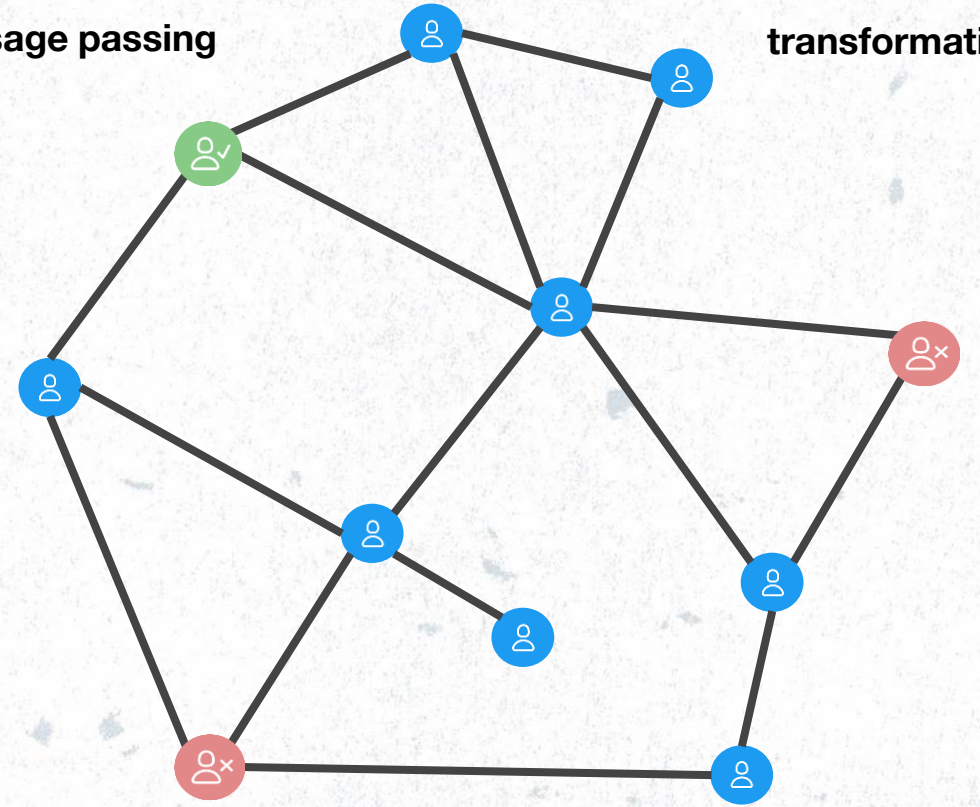
Message passing neural network as diffusion of information on a graph

input

message passing

transformation

output



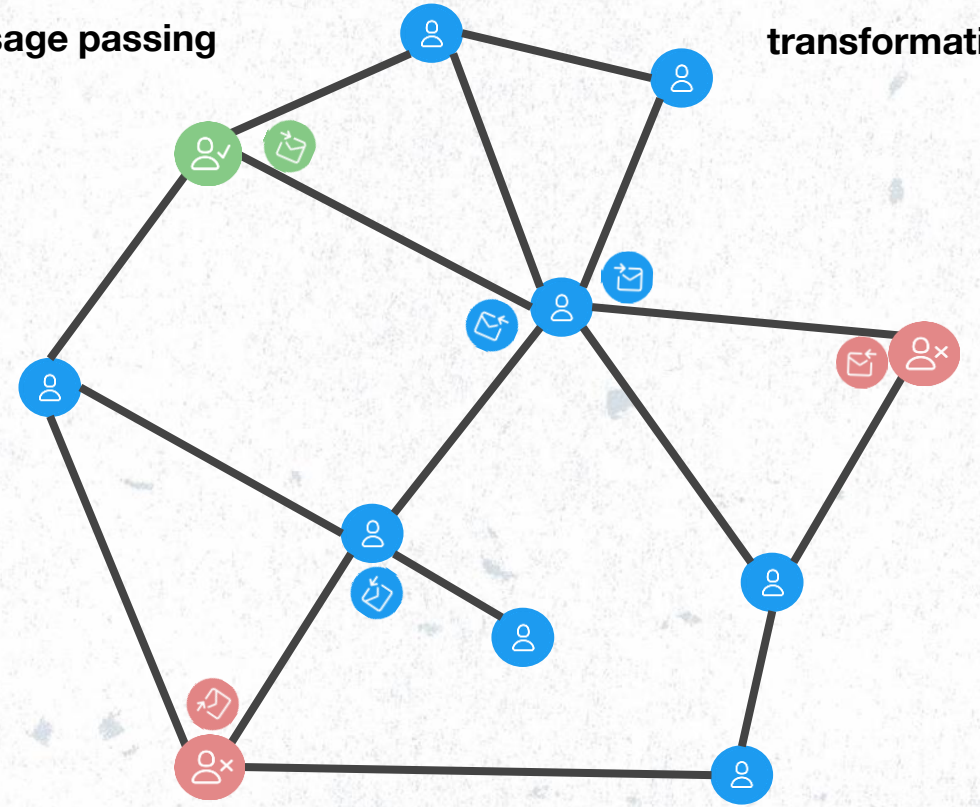
Message passing neural network as diffusion of information on a graph

input

message passing

transformation

output



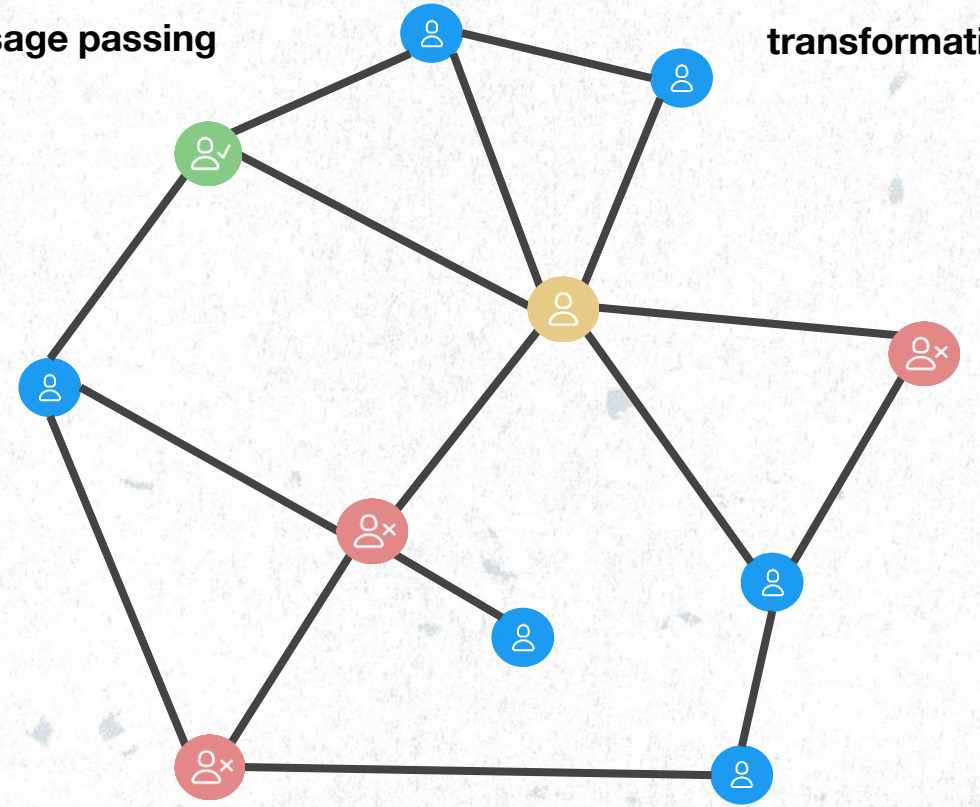
Message passing neural network as diffusion of information on a graph

input

message passing

transformation

output



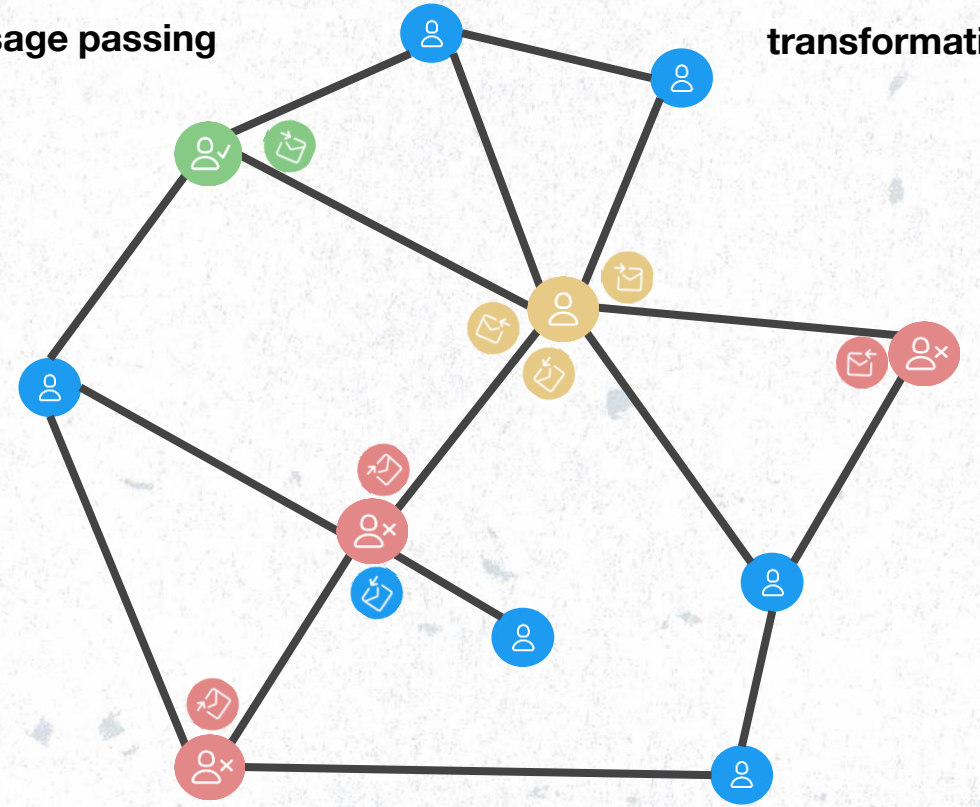
Message passing neural network as diffusion of information on a graph

input

message passing

transformation

output



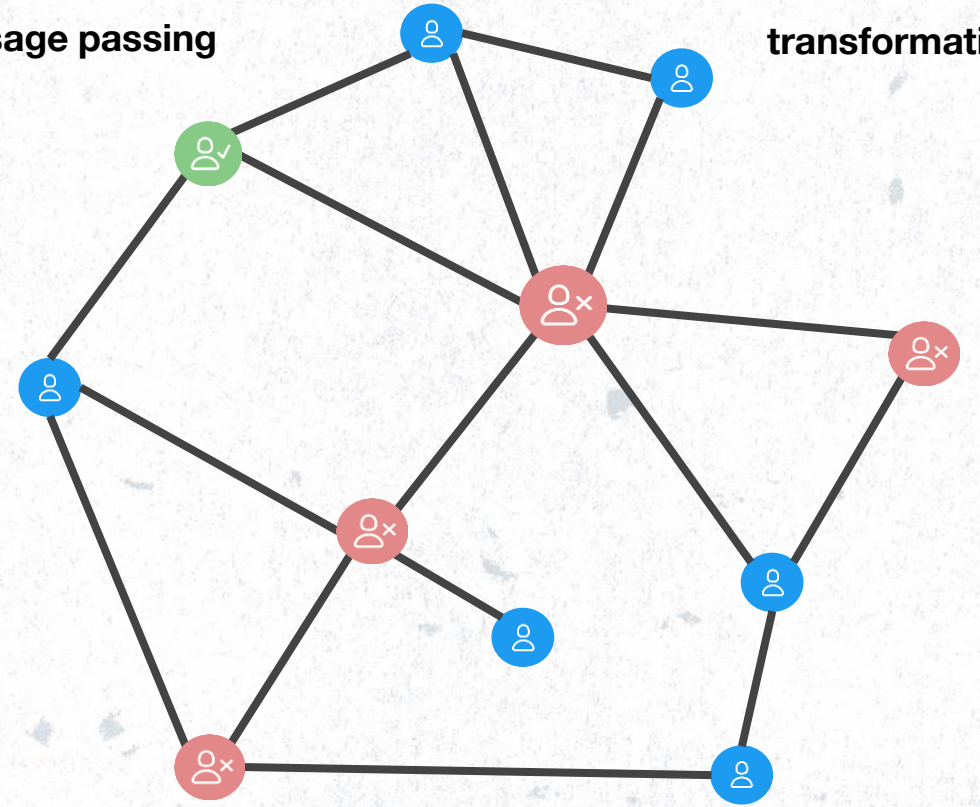
Message passing neural network as diffusion of information on a graph

input

message passing

transformation

output



Spatial discretisation

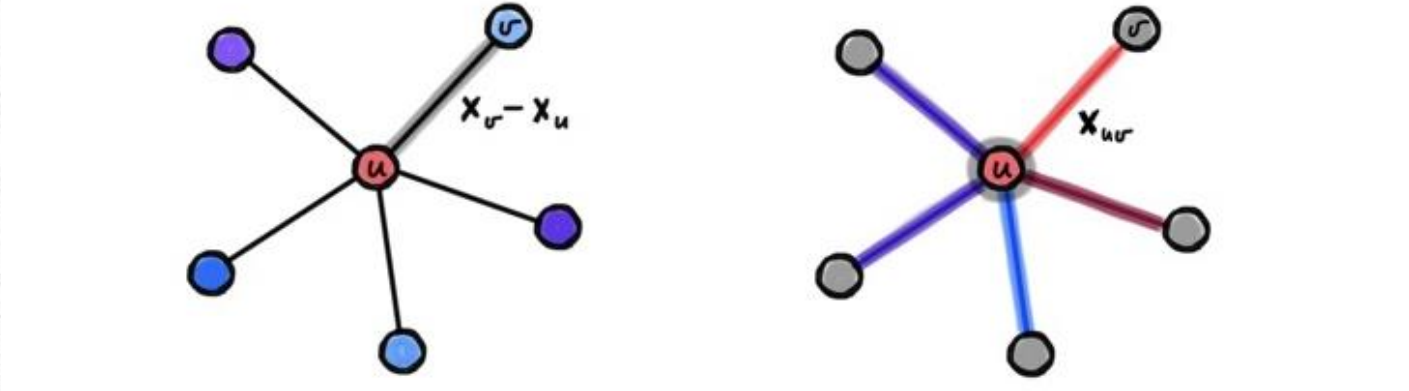
$$\frac{\partial x(u,t)}{\partial t} = \mathrm{div}[\nabla x(u,t)] \qquad (1)$$

gradient - flow along edges

$$(\nabla X)_{uv} = x_u - x_v$$

divergence - aggregation of edges

$$\big(\mathrm{div}(X)\big)_u = \sum_v w_{uv} x_{uv}$$

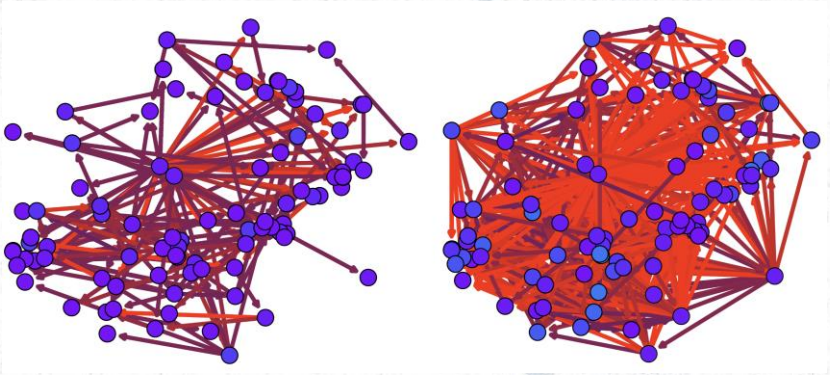


$$\dot{X}(t) = \big(A\big(X(t)\big) - I\big)X(t) \quad (2)$$

Graph Rewiring

Decouple input graph from information propagation graph

- **Neighbourhood sampling (GraphSAGE)**[1]
- **Multi-hop filters (SIGN)**[2]
- **Complete graph**[3]
- **Topology diffusion (DIGL)**[4]
- **Learnable graph (Dynamic Graph CNN)**[5]



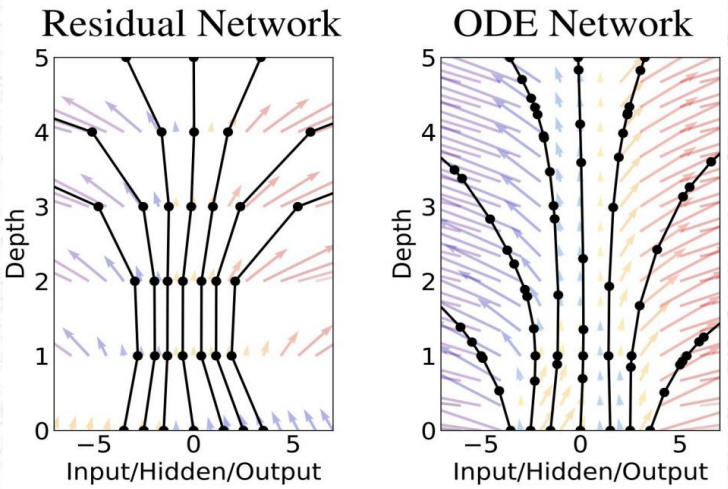
[1]Hamilton et al. 2017; [2]Rossi, Frasca, et B. 2020; [3]Alon, Yahav 2020; [4]Klicpera et al. 2019; [5]Wang et B 2018; Kazi, Cosmo, et B. 2020

Temporal discretisation

Neural ODEs:

ResNet: $\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + f(\boldsymbol{h}_t, \theta_t)$

Dynamics: $\frac{d\boldsymbol{h}(t)}{dt} = f(\boldsymbol{h}_t, \theta_t, t)$



ODE Solvers:

$$\boldsymbol{z}(t_1) = \boldsymbol{z}(t_0) + \int_{t_0}^{T} f(\boldsymbol{z}(t), t, \theta)\, dt = \text{ODESolve}\,(\boldsymbol{z}(t_0), f, t_0, t_1, \theta)$$

Chen et al. 2019; Haber et al. 2019

PDEs are intimately related to GNNs, so what do we get:

- New perspectives on old problems like over-smoothing
- New architectures inspired by numerical ODE solvers
- Stability conditions
- Access to new domains inspired by PDEs

Diffusion in image processing

Two perspectives on image processing continuous and discrete

Continuous:

– the real world is continuous as are high definition images

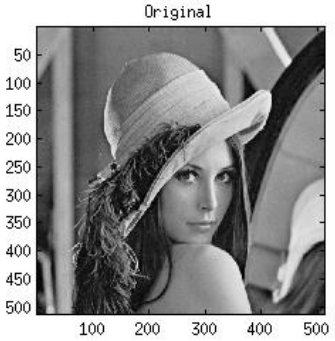
Treat an image as a function, evolve it using PDEs

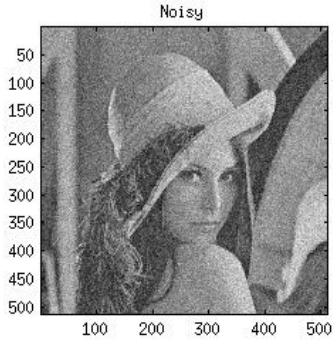
Discrete:

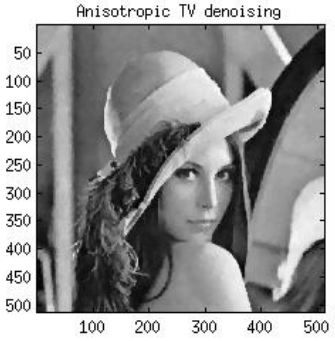
– a computer will always model an image as a tensor

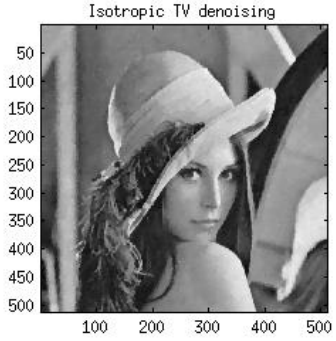
Treat an image as a tensor and evolve it using linear algebra

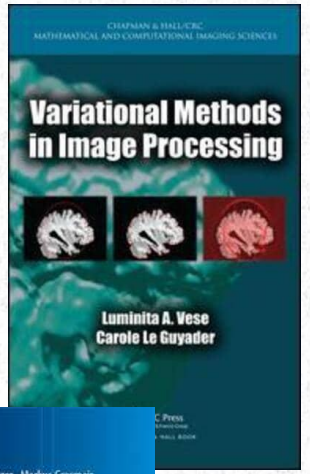
Variational methods in image processing

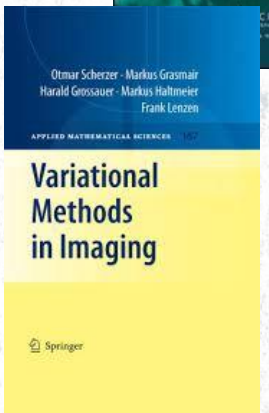










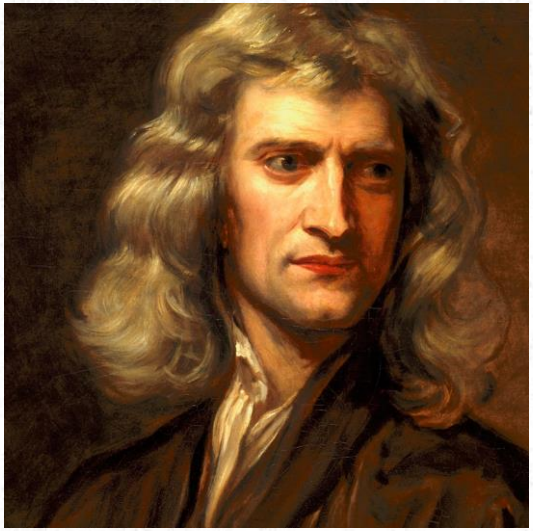


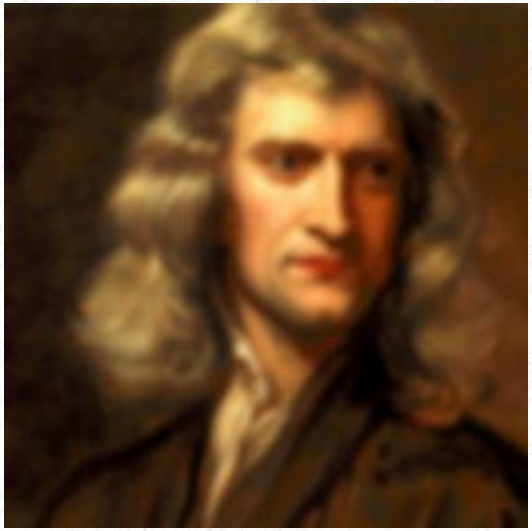
Perona, Malik 1990; Sochen et al. 1998; Tomasi, Manduchi 1998; Weickert 1998; Buades et al. 2005

Diffusion in Image Processing

$$\frac{\partial}{\partial t}x = c\Delta x$$

$$\frac{\partial}{\partial t}x = \mathrm{div}\left[\frac{1}{1 + (|\nabla x|/\lambda)^2}\nabla \boldsymbol{x}\right]$$





Homogeneous diffusion

Non-homogeneous diffusion

Perona, Malik 1990; Sochen et al. 1998; Tomasi, Manduchi 1998; Weickert 1998; Buades et al. 2005

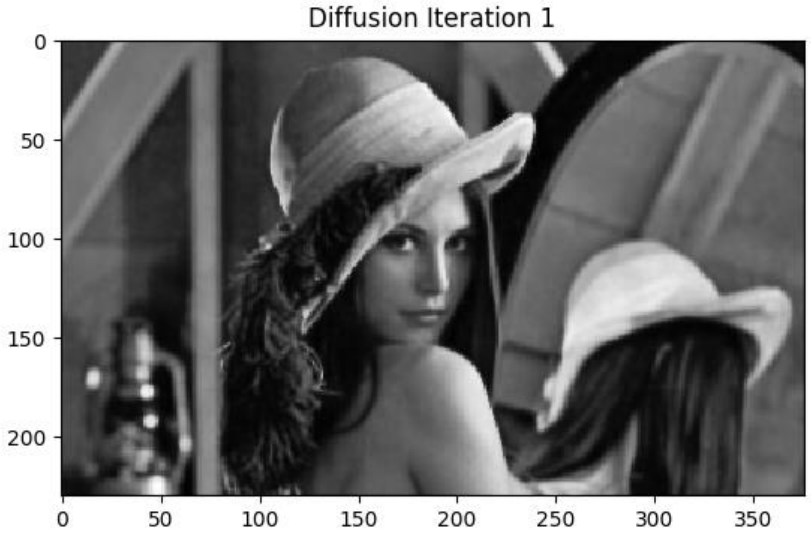
Variational methods in image processing – 50 frame diffusion

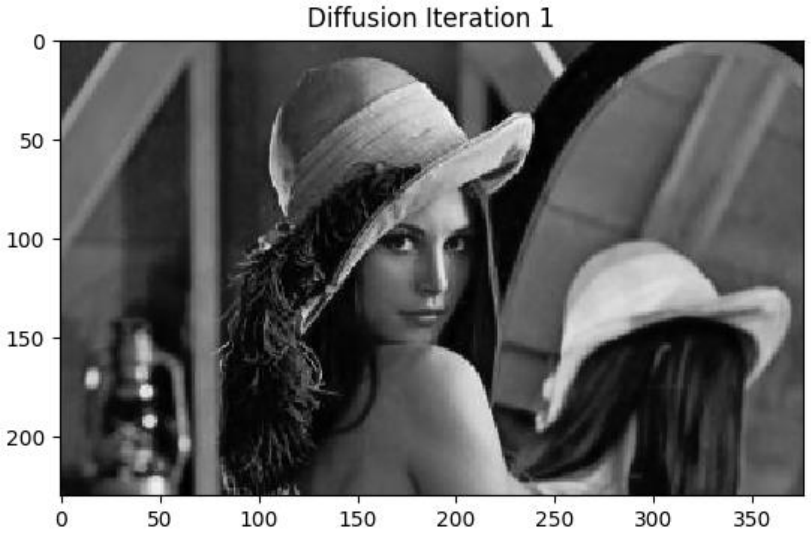
Gaussian:

$$g(x) = c$$

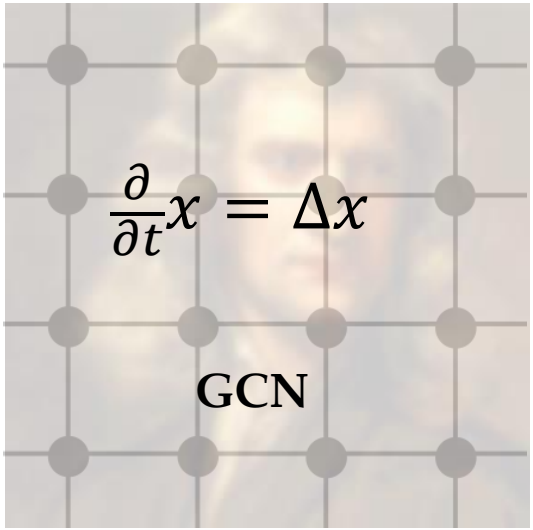
Perona & Malik:

$$g(|\nabla x|) = \frac{1}{1 + (|\nabla x|/\lambda)^2}$$

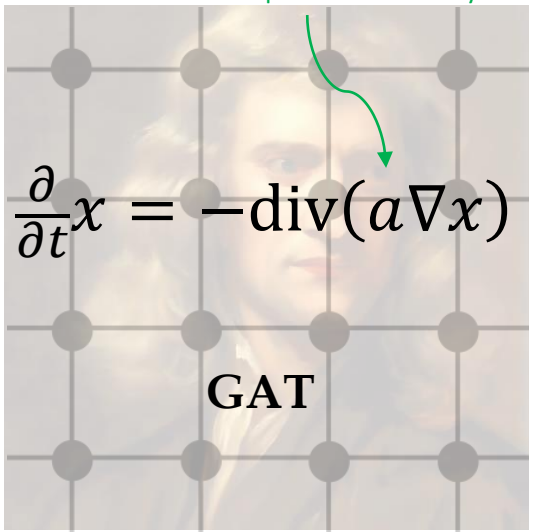




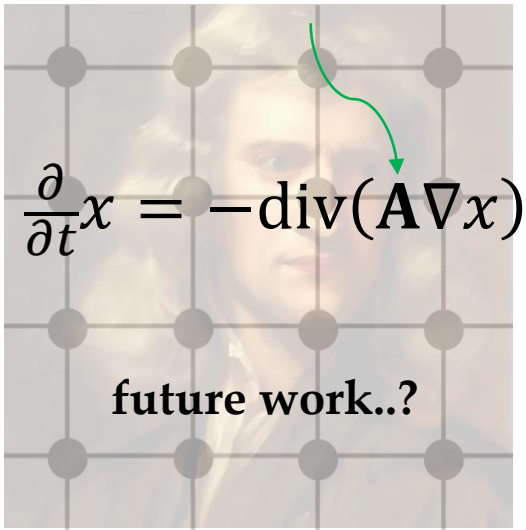
Attentional diffusivity for GNNs



Position-dependent diffusivity



Position & direction dependent diffusivity



Fick 1855

GRAND (ICML 2021):

GRAND: Graph Neural Diffusion

Benjamin P. Chamberlain [*1] **James Rowbottom** [*1] **Maria Gorinova** [1] **Stefan Webb** [1] **Emanuele Rossi** [1]
Michael M. Bronstein [1 2 3]

Abstract

We present Graph Neural Diffusion (GRAND) that approaches deep learning on graphs as a continuous diffusion process and treats Graph Neural Networks (GNNs) as discretisations of an underlying PDE. In our model, the layer structure and topology correspond to the discretisation choices of temporal and spatial operators. Our approach allows a principled development of a broad new class of GNNs that are able to address the common plights of graph learning models such as depth, oversmoothing, and bottlenecks. Key to the success of our models are stability with respect to perturbations in the data and this is addressed for both implicit and explicit discretisation schemes. We develop linear and nonlinear versions of GRAND, which achieve competitive results on many standard graph benchmarks.

PDEs are among the most studied mathematical constructions, with a vast literature dating back at least to Leonhard Euler in the eighteenth century. This includes various discretisation schemes, numerical methods for approximate solutions, and theorems for their existence and stability. Historically, PDE-based methods have been used extensively in signal and image processing (Perona & Malik, 1990), computer graphics (Sun et al., 2009), and more recently, in machine learning (Chen et al., 2018).

Our goal is to show that the tools of PDEs can be used to understand existing GNN architectures and as a principled way to develop a broad class of new methods. We focus on GNN architectures that can be interpreted as information diffusion on graphs, modelled by the diffusion equation. In doing so, we show that many popular GNN architectures can be derived from a single mathematical framework by different choices of the form of diffusion equation and discretisation schemes. Standard GNNs are equivalent to the explicit single-step Euler scheme that is inefficient and

New architectures

Simple GNN by discretising time in the graph diffusion equation:

$$\dot{\boldsymbol{X}}(t) = \left(\boldsymbol{A}\left(\boldsymbol{X}(t)\right) - \boldsymbol{I}\right)\boldsymbol{X}(t)\ \Big) \quad (2)$$

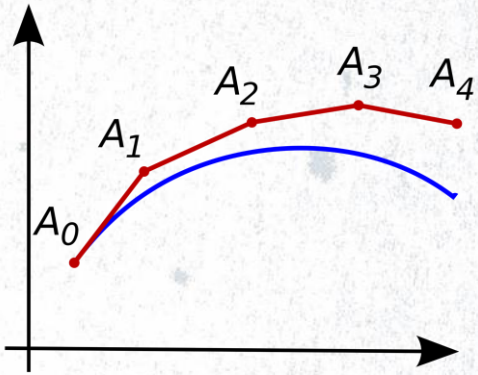
Explicit Euler temporal discretisation

$$\boldsymbol{X}(k+1) = \boldsymbol{X}(k) + \tau\left[\boldsymbol{A}\left(\boldsymbol{X}(k)\right) - \boldsymbol{I}\right]\boldsymbol{X}(k)$$

Set time step τ = 1 get simplified GCN

$$\boldsymbol{X}(k+1) = \boldsymbol{A}\left(\boldsymbol{X}(k)\right)\boldsymbol{X}(k)$$

without feature channel mixing and non-linearities



Kipf 2017

Better ODE Solvers

Multi-step methods:

$$x_{n+1} + \sum_{i=1}^{s} \alpha_i x_{n+1-i} = \tau \sum_{i=0}^{s} \beta_i f_{n+1-i}$$

Adaptive step-size solvers:

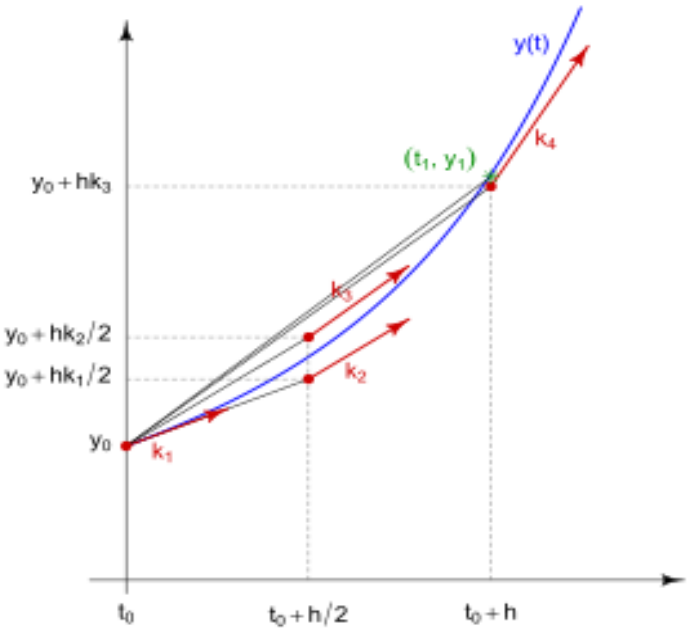
order p: $x_p(\tau)$

order p-1: $x_{p-1}(\tau)$

error: $\varepsilon = x_p(\tau) - x_{p-1}(\tau)$

tolerance: $etol = atol + rtol * \max(|x_0|, |x_1|)$

Eg: Runge-Kutta 4



If $\varepsilon > etol$

then decrease step-size τ

Implicit Solvers

Implicit Versus Explicit Euler's Methods

- **Explicit Euler's Method:**

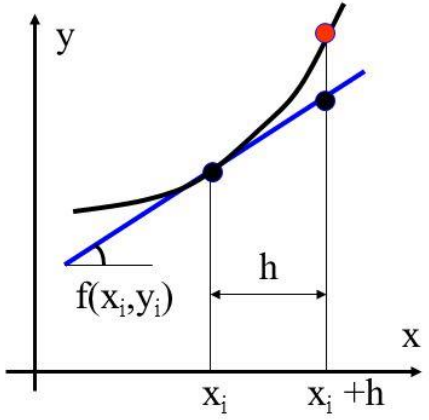
$$\frac{dy}{dx} = f(x, y)$$

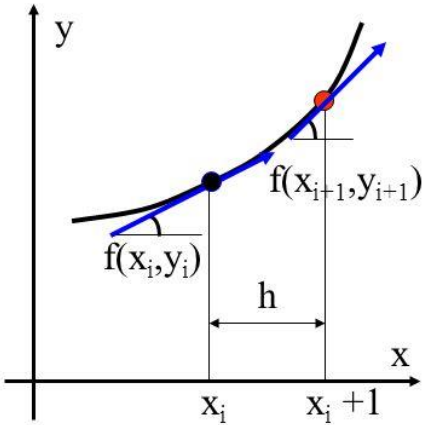
$$y_{i+1} = y_i + f(x_i, y_i) \cdot h$$

- **Implicit Euler's Method:**

$$\frac{dy}{dx} = f(x, y)$$

$$y_{i+1} = y_i + f(x_{i+1}, y_{i+1}) \cdot h$$

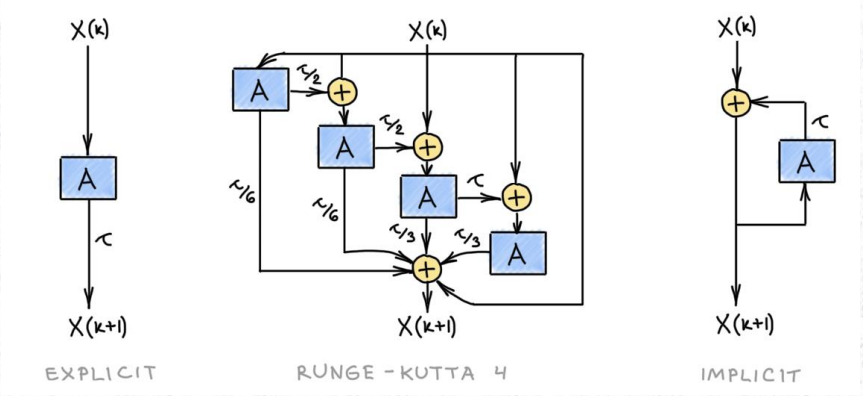




Explicit ODE Schemes:

$$\frac{x^{(k+1)} - x^{(k)}}{\tau} = \bar{A}\left(x^{(k)}\right)x^{(k)}$$

$$x^{(k+1)} = x^{(k)} + \tau\bar{A}\left(x^{(k)}\right)x^{(k)}$$

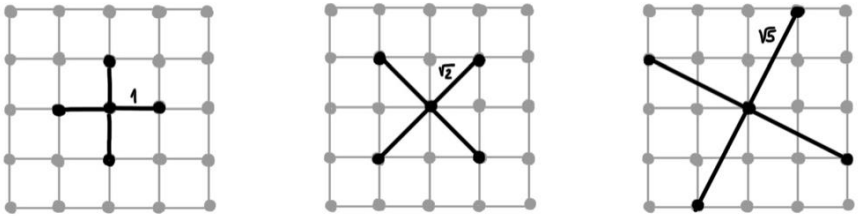


(Semi)-Implicit ODE Schemes:

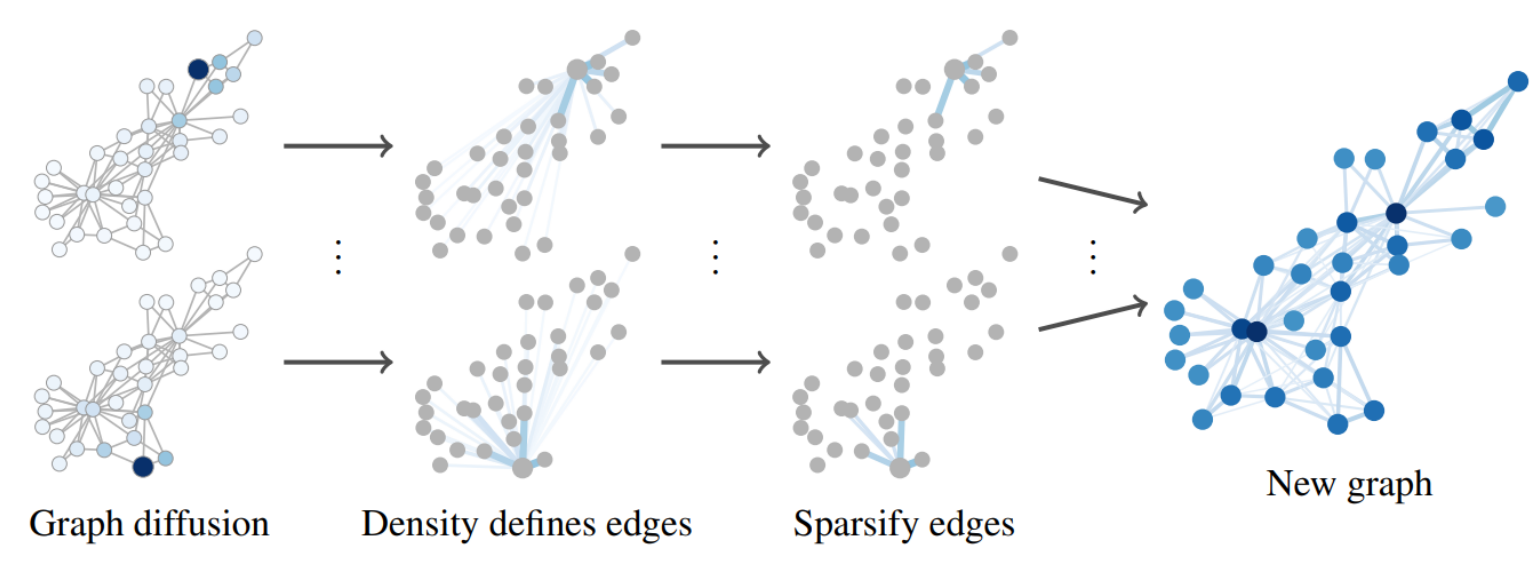
$$\frac{x^{(k+1)} - x^{(k)}}{\tau} = \bar{A}\left(x^{(k)}\right)x^{(k+1)}$$

$$\left[I - \tau\bar{A}\left(x^{(k)}\right)\right]x^{(k+1)} = x^{(k)}$$

$$x^{(k+1)} = B\left(x^{(k)}\right)^{-1}x^{(k)}$$



Rewiring – Diffusion Improves Graph Learning



Klicpera et al. 2019

Threshold attention: $\mathcal{E}' = \{(i,j): (i,j) \in \mathcal{E} \text{ and } a_{ij} > \rho\}$

Stability

If you treat a GNN as a PDE you can explain instability

Well posedness of the differential equation:

ε-δ definition:

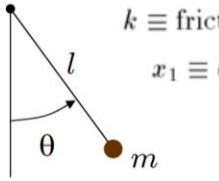
Given any $\varepsilon > 0$
$\exists\, \delta > 0$:
$|x(t) - \hat{x}(t)| < \varepsilon$
if $|x(0) - \hat{x}(0)| < \delta$

Robustness of the machine learning problem:

"small perturbations in the input data.."

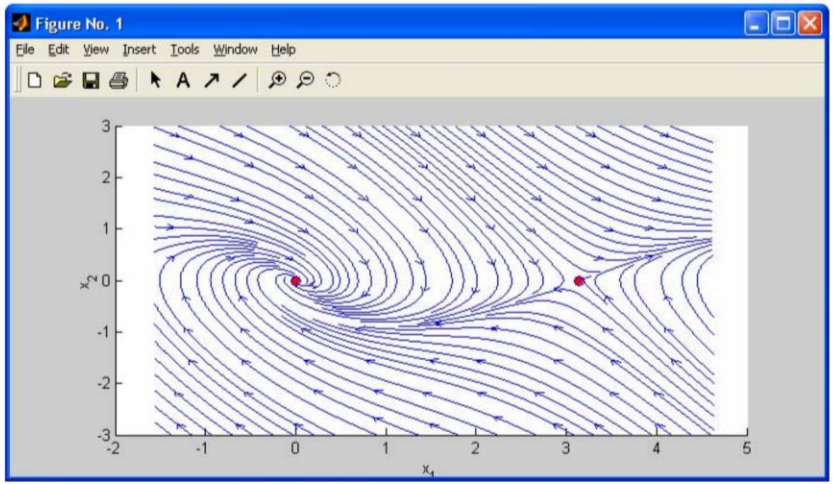
Example #1: Pendulum

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\frac{g}{l}\sin x_1 - \frac{k}{m}x_2$$



$k \equiv$ friction coefficient

$x_1 \equiv \theta \qquad x_2 \equiv \dot{\theta}$



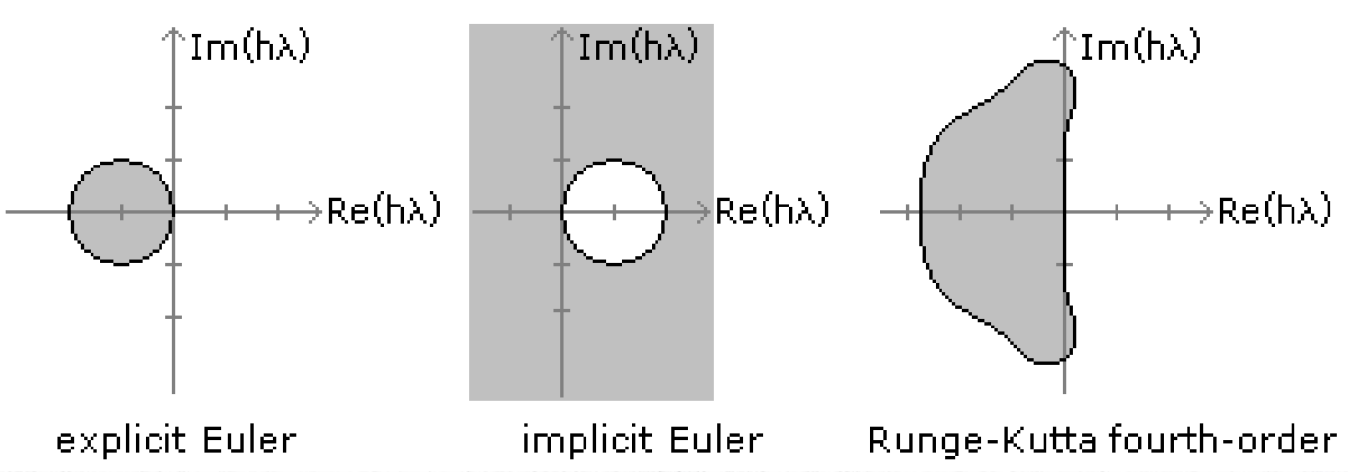
x_{eq}=(0,0) stable

x_{eq}=(π,0) unstable

pend.m

Stability results for the numerical approximations:

Linking step size and eigenvalues



David Eberly

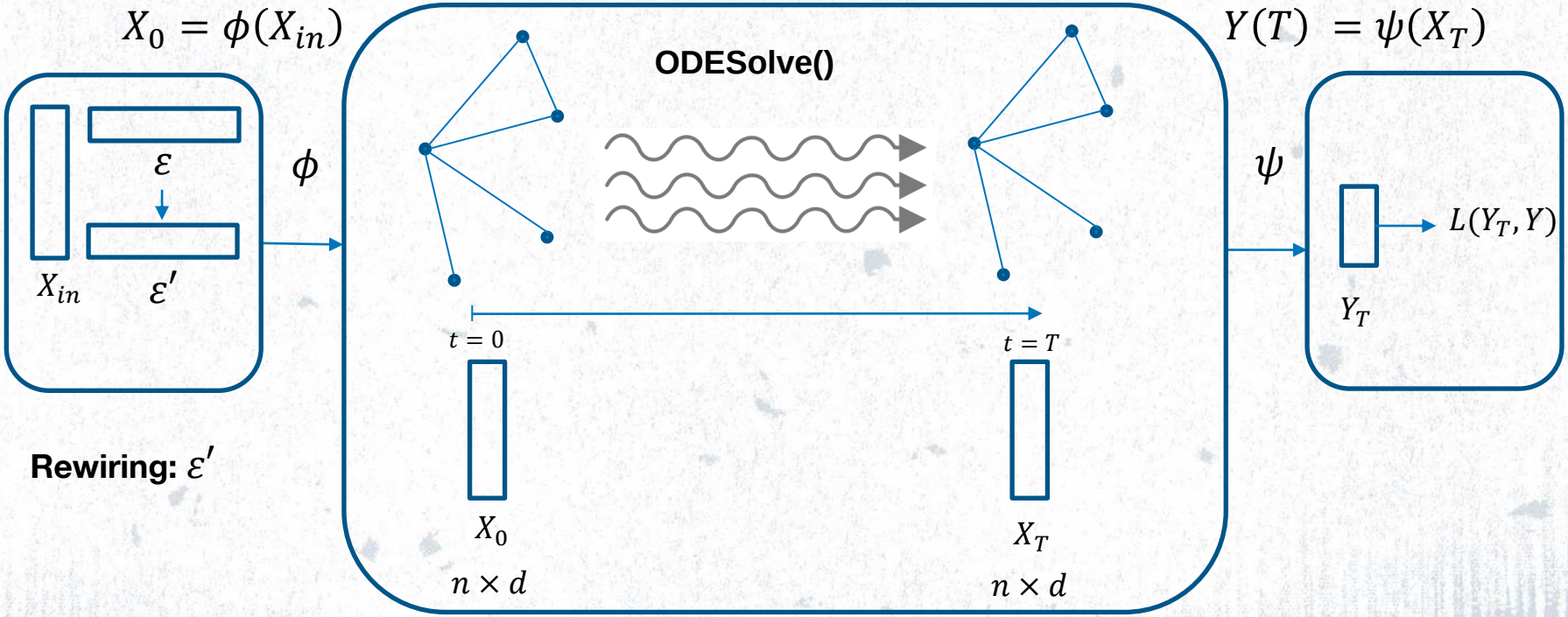
GRAND has special case results due to stochastic attention matrix A

GRAND architecture:

Layers: $k = \frac{T}{\tau}$

Encoder:

$X_0 = \phi(X_{in})$



Decoder:

$Y(T) = \psi(X_T)$

Rewiring: ε'

The features are evolved via the system of ODEs

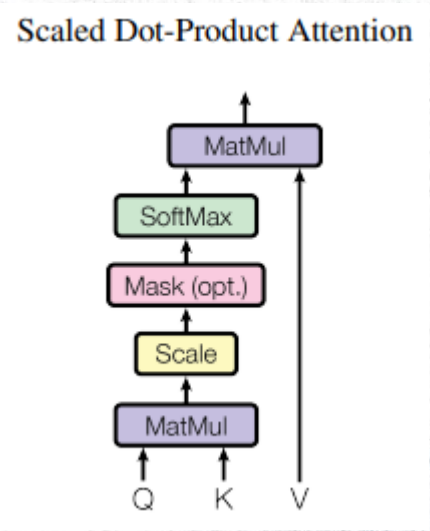
$$\boldsymbol{X}(T) = \psi\left(\phi(\boldsymbol{X}_{in}) + \int_0^T \frac{\partial \boldsymbol{X}(t)}{\partial t} dt\right) = \text{ODESolve}\,(\boldsymbol{X}_0, f, t_0, T, \theta)$$

the dynamics are learnt via the diffusivity $a(,)$

$$f = \frac{\partial}{\partial t}\boldsymbol{X}(t) = \big(\boldsymbol{A}\big(\boldsymbol{X}(t)\big) - \boldsymbol{I}\big)\boldsymbol{X}(t)$$

We use scaled dot-product attention

$$\boldsymbol{A}_{ij} = a\big(\boldsymbol{X}_i, \boldsymbol{X}_j\big) = softmax\left(\frac{(W_K \boldsymbol{X}_i)^\top W_Q \boldsymbol{X}_j}{\sqrt{d_k}}\right)$$



Vaswani et al. 2017

Three types of GRAND:

GRAND linear:

attention from initial conditions

$$\frac{d\boldsymbol{X}(t)}{dt} = \left(\boldsymbol{A}\left(\boldsymbol{X}(0)\right) - \boldsymbol{I}\right)\boldsymbol{X}(t)$$

$$\boldsymbol{X}(t) = e^{\bar{\boldsymbol{A}}t}\boldsymbol{X}(0)$$

GRAND non-linear:

attention recalculated at each step in solver

$$\frac{d\boldsymbol{X}(t)}{dt} = \left(\boldsymbol{A}\left(\boldsymbol{X}(t)\right) - \boldsymbol{I}\right)\boldsymbol{X}(t)$$

GRAND non-linear with rewiring:

decouple the given and computational graph

$$\mathcal{E}' = \{(i,j): (i,j) \in \mathcal{E} \text{ and } a_{ij} > \rho\}$$

GRAND depth analysis:

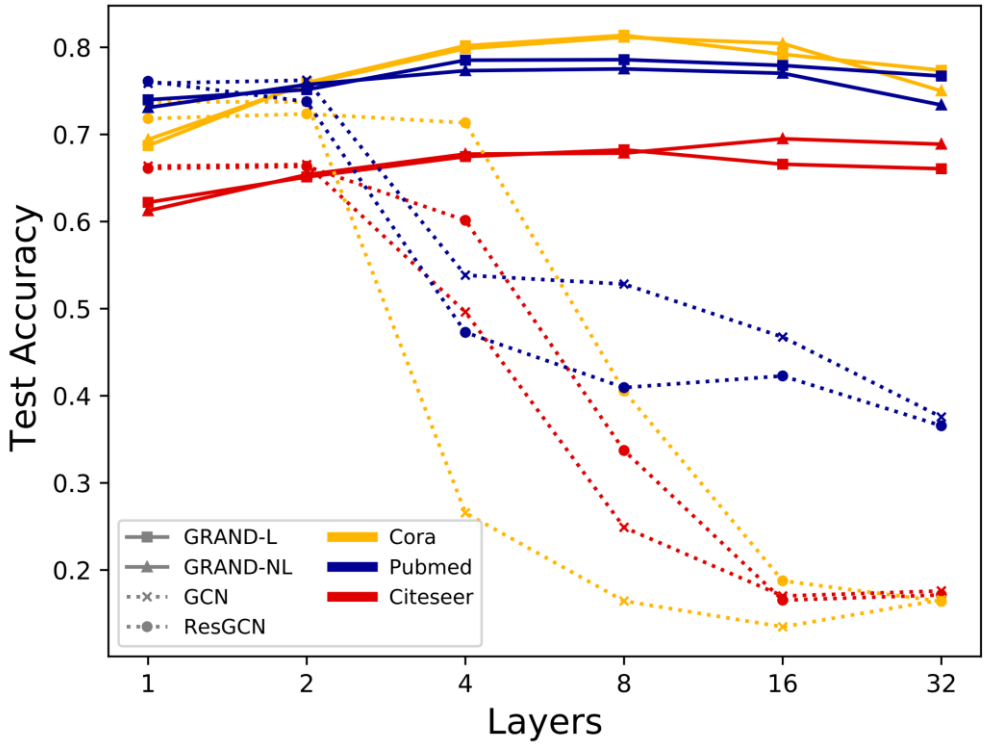
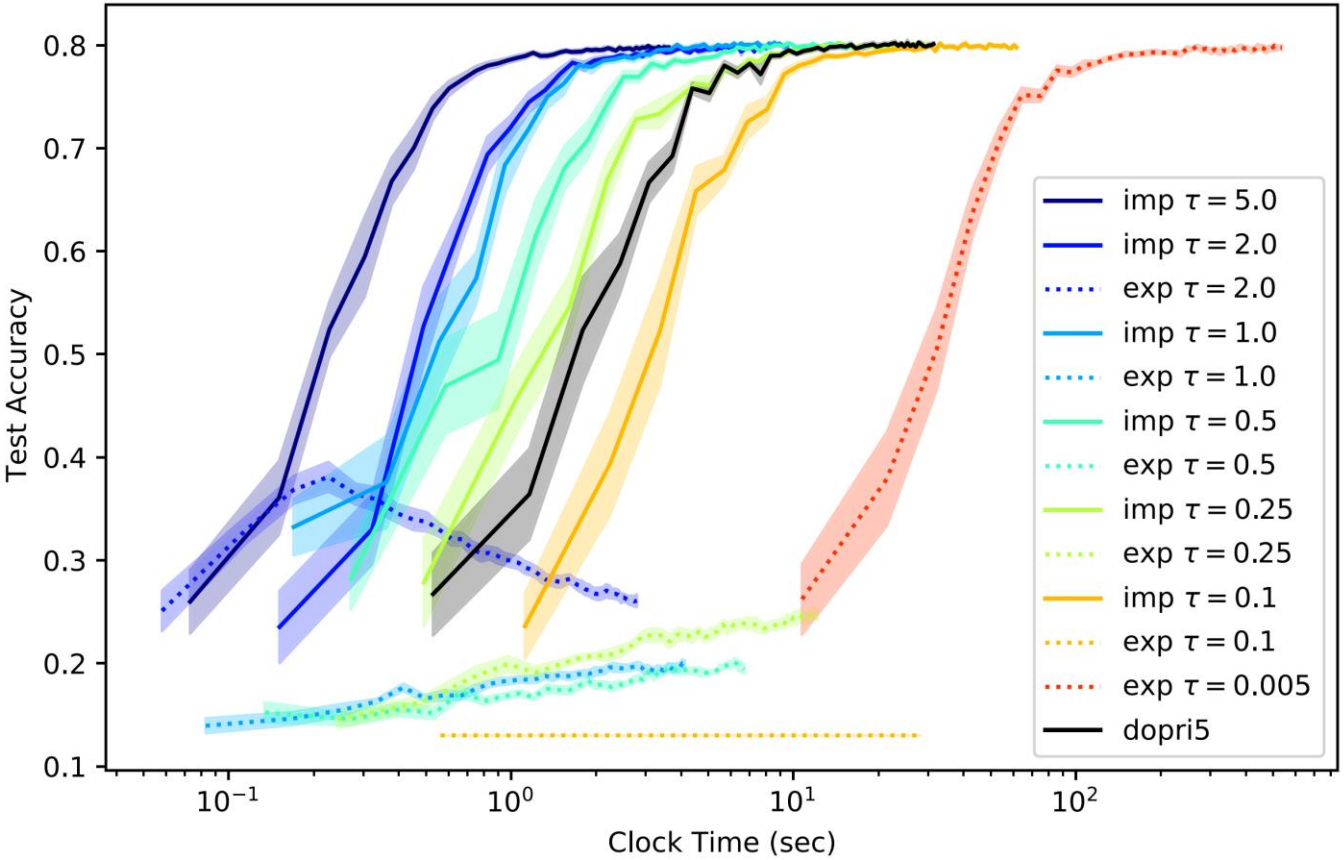


Figure 2. Performance of architectures of different depth.

GRAND: implicit and explicit solvers



Advantages and disadvantages of GRAND

Advantages:

- Learned diffusivity enables edge detection allowing for a deeper more expressive network
- Advanced ODE solvers reduces error, improves stability and offers new GNN architectures

Disadvantages :

- The spatial discretisation is heuristic with apriori graph rewiring and attention thresholding

We would like a more principled way to rewire the graph and spatially discretise the PDE

BLEND (NeurIPS 2021):

Beltrami Flow and Neural Diffusion on Graphs

Benjamin P. Chamberlain*
Twitter Inc.
bchamberlain@twitter.com

James Rowbottom*
Twitter Inc.

Davide Eynard
Twitter Inc.

Francesco Di Giovanni
Twitter Inc.

Xiaowen Dong
University of Oxford

Michael M. Bronstein
Twitter Inc. and Imperial College London

Abstract

We propose a novel class of graph neural networks based on the discretised *Beltrami flow*, a non-Euclidean diffusion PDE. In our model, node features are supplemented with positional encodings derived from the graph topology and jointly evolved by the Beltrami flow, producing simultaneously continuous feature learning and topology evolution. The resulting model generalises many popular graph neural networks and achieves state-of-the-art results on several benchmarks.

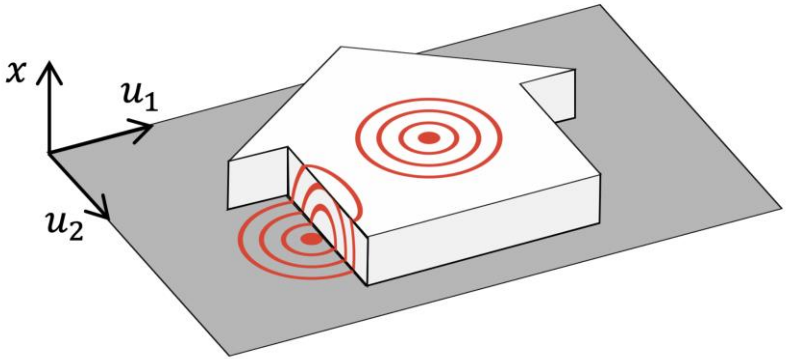
1 Introduction

The majority of graph neural networks (GNNs) are based on the message passing paradigm [30], wherein node features are learned by means of a non-linear propagation on the graph. Multiple recent works have pointed to the limitations of the message passing approach. These include; limited expressive power [7, 9, 80, 95], the related oversmoothing problem [60, 62] and the bottleneck phenomena [1, 93], which render such approaches inefficient, especially in deep GNNs. Multiple alternatives have been proposed, among which are higher-order methods [7, 54] and decoupling the propagation and input graphs by modifying the topology, often referred to as *graph rewiring*. Topological modifications can take different forms such as graph sampling [32], kNN [43], using the complete graph [1, 86], latent graph learning [36, 89], or multi-hop filters [73, 92]. However, there is no agreement in the literature on when and how to modify the graph, and a single principled framework for doing so.

Images as embedded manifolds

$$\frac{\partial}{\partial t}\mathbf{x} = -\mathrm{div}(a(\mathbf{x})\nabla\mathbf{x})$$

Non-linear diffusion



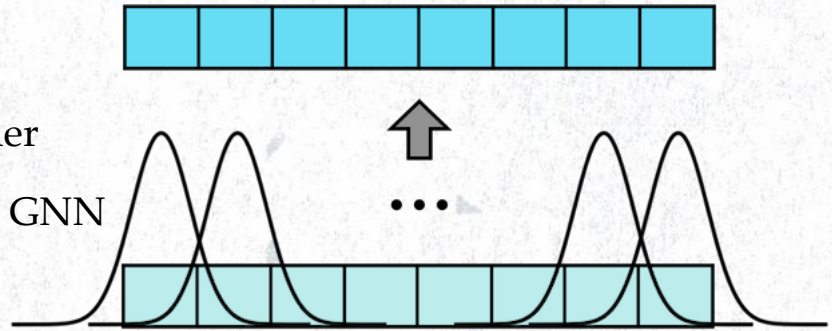
$$\frac{\partial}{\partial t}\mathbf{z} = \Delta_{\mathbf{G}}\mathbf{z}$$

Non-Euclidean diffusion

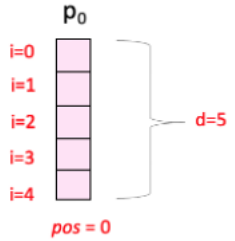
Sochen et al. 1998

Intro to positional encodings

- An idea that was first popularized in the transformer
- Now widely used to increase the expressiveness of GNN
- Many options
  - Random node features[1]
  - Graph Laplacian eigenvectors[2]
  - Graph substructure counts[3]
  - Bags of subgraphs[4]



$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$



[1]Sato et al. 2020; [2]Vaswani et al. 2017; Qiu et al. 2020; Dwivedi et al. 2020; [3]Bouritsas, Frasca, et B. 2020; [4]Bevilacqua, Frasca, Lim, et B., Maron 2021

What to do with the positional encodings

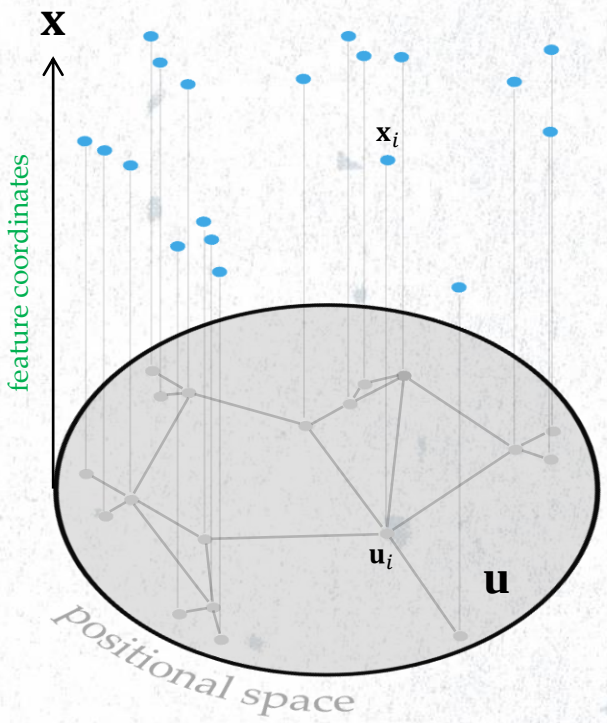
- In computer vision they are just for edge detection and then thrown away
- The graph approach is more elegant as the evolved positional encodings can be used to rewire the graph?
- Why rewire the graph?
  - Already popular to do decouple the given and computational graph in the GNN literature

[1]Hamilton et al. 2017; [2]Rossi, Frasca, et B. 2020; [3]Alon, Yahav 2020; [4]Klicpera et al. 2019; [5]Wang et B 2018; Kazi, Cosmo, et B. 2020

Graph Beltrami flow

- Graph with positional and feature node coordinates $\mathbf{z}_i = (\mathbf{u}_i, \mathbf{x}_i)$
- **Graph Beltrami flow**

$$\frac{\partial}{\partial t}\mathbf{z}_i = \sum_{j:(i,j)\in E} a(\mathbf{z}_i, \mathbf{z}_j)(\mathbf{z}_j - \mathbf{z}_i)$$



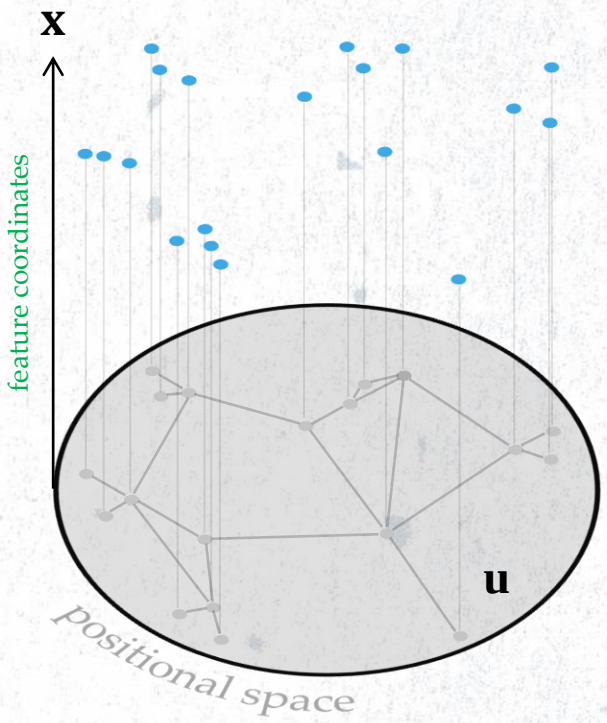
Chamberlain et B. 2021

Graph Beltrami flow

- Graph with positional and feature node coordinates $\mathbf{z}_i = (\mathbf{u}_i, \mathbf{x}_i)$
- **Graph Beltrami flow**

$$\frac{\partial}{\partial t}\mathbf{z}_i = \sum_{j:(i,j)\in E} a(\mathbf{z}_i, \mathbf{z}_j)(\mathbf{z}_j - \mathbf{z}_i)$$

  - Evolution of $\mathbf{x}$ = feature diffusion



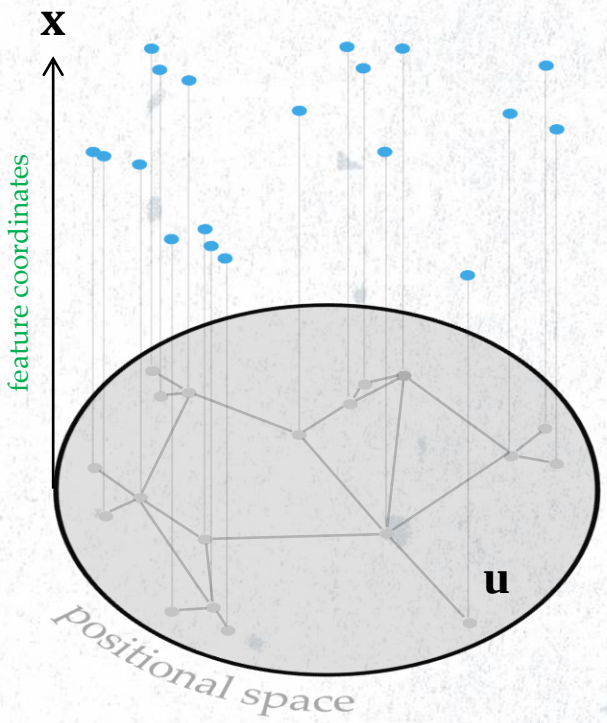
Chamberlain et B. 2021

Graph Beltrami flow

- Graph with positional and feature node coordinates $\mathbf{z}_i = (\mathbf{u}_i, \mathbf{x}_i)$
- **Graph Beltrami flow**

$$\frac{\partial}{\partial t}\mathbf{z}_i = \sum_{j:(i,j)\in E} a(\mathbf{z}_i, \mathbf{z}_j)(\mathbf{z}_j - \mathbf{z}_i)$$

  - Evolution of $\mathbf{x}$ = feature diffusion



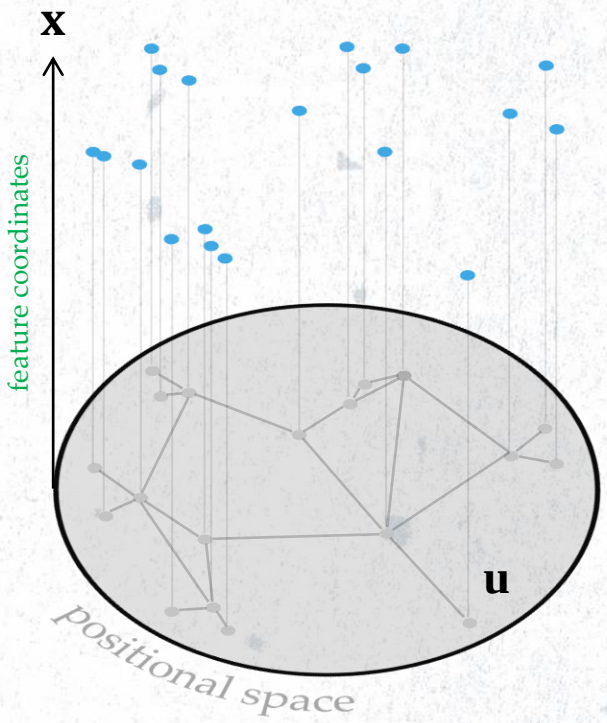
Chamberlain et B. 2021

Graph Beltrami flow

- Graph with positional and feature node coordinates $\mathbf{z}_i = (\mathbf{u}_i, \mathbf{x}_i)$
- **Graph Beltrami flow**

$$\frac{\partial}{\partial t}\mathbf{z}_i = \sum_{j:(i,j)\in E} a(\mathbf{z}_i, \mathbf{z}_j)(\mathbf{z}_j - \mathbf{z}_i)$$

  - Evolution of $\mathbf{x}$ = feature diffusion
  - Evolution of $\mathbf{z}$ = graph rewiring



Chamberlain et B. 2021

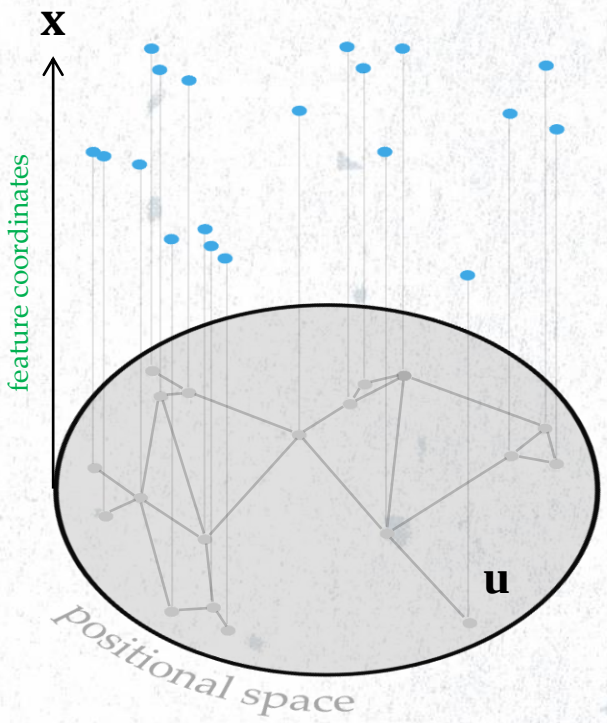
Graph Beltrami flow

- Graph with positional and feature node coordinates $\mathbf{z}_i = (\mathbf{u}_i, \mathbf{x}_i)$
- **Graph Beltrami flow**

$$\frac{\partial}{\partial t}\mathbf{z}_i = \sum_{j:(i,j)\in E'} a(\mathbf{z}_i, \mathbf{z}_j)(\mathbf{z}_j - \mathbf{z}_i)$$

rewired graph

  - Evolution of $\mathbf{x}$ = feature diffusion
  - Evolution of $\mathbf{z}$ = graph rewiring



Chamberlain et B. 2021

GNN Architectures as instances of BLEND

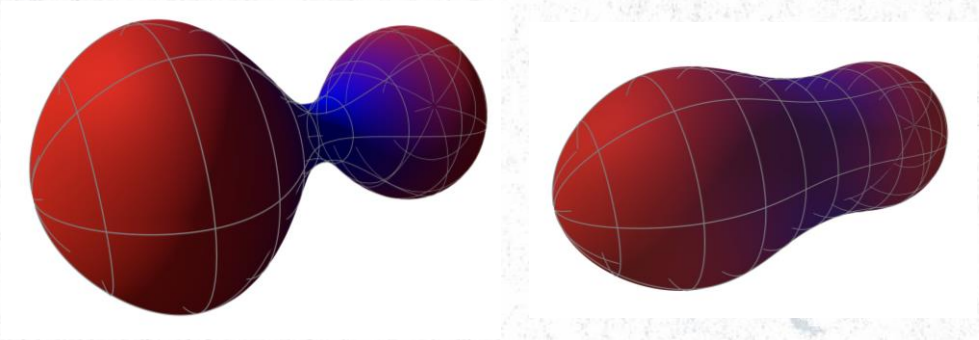
$$\mathbf{Z}^{(k+1)} = \mathbf{Q}\left(\mathbf{Z}^{(k)}\right)\mathbf{Z}^{(k)}$$

Method	Evolution	Diffusivity	Graph	Discretisation
ChebNet	$\mathbf{X}$	Fixed $\mathbf{A}$	Fixed E	Explicit Fixed step
GAT	$\mathbf{X}$	$\mathbf{A}(\mathbf{X})$	Fixed E	Explicit Fixed step
MoNet	$\mathbf{X}$	$\mathbf{A}(\mathbf{U})$	Fixed E	Explicit Fixed step
Transformer	$\mathbf{X}$	$\mathbf{A}(\mathbf{U},\mathbf{X})$	Fixed $E = V^2$	Explicit Fixed step
DIGL	$\mathbf{X}$	$\mathbf{A}(\mathbf{X})$	Fixed $E(\mathbf{U})$	Explicit Fixed step
DGCNN	$\mathbf{X}$	$\mathbf{A}(\mathbf{X})$	Adaptive $E(\mathbf{X})$	Explicit Fixed step
BLEND	$\mathbf{U},\mathbf{X}$	$\mathbf{A}(\mathbf{U},\mathbf{X})$	Adaptive $E(\mathbf{U})$	Explicit Adaptive step / Implicit

Ricci flow

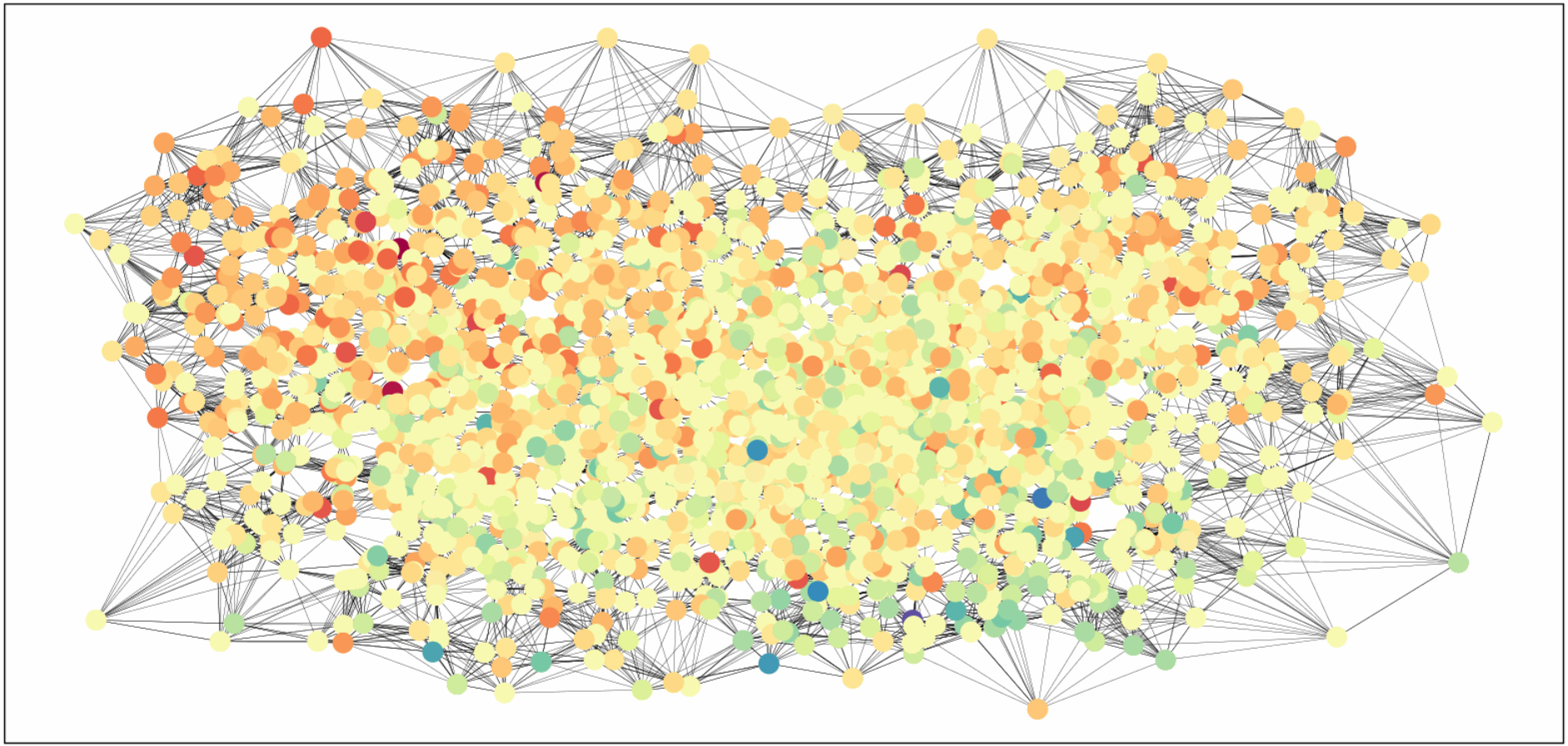
- **Ricci flow:** "diffusion of the Riemannian metric"

$$\frac{\partial g_{ij}}{\partial t} = R_{ij}$$

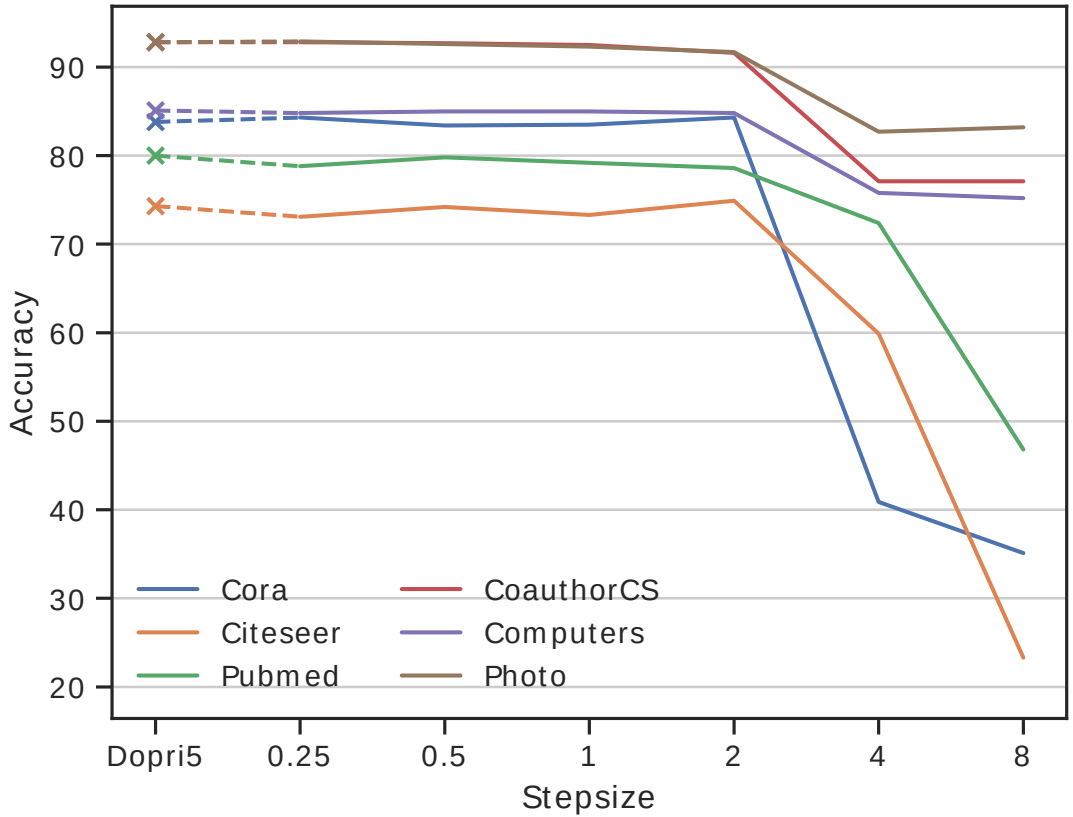


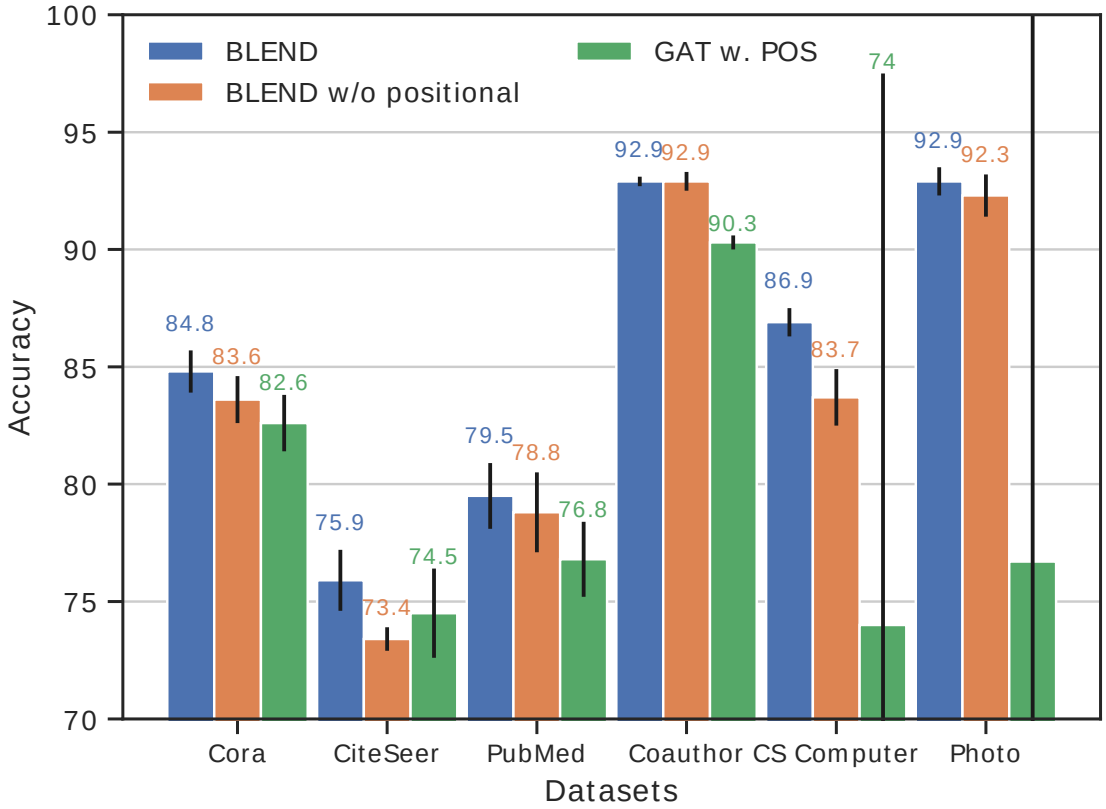
Evolution of a manifold under Ricci flow

Ricci 1903; Hamilton 1988; Perelman 2003


Beltrami Flow, diffusion time=0

Method	CORA	CiteSeer	PubMed	Coauthor CS	Computer	Photo	ogb-arxiv
GCN	81.5±1.3	71.9 ±1.9	77.8±2.9	91.1±0.5	82.6±2.4	91.2±1.2	71.74±0.29
GAT	81.8±1.3	71.4±1.9	78.7±2.3	90.5±0.6	78.0±19	85.7±20	**73.01**±0.19*
GAT-ppr	81.6±0.3	68.5±0.2	76.7±0.3	91.3±0.1	85.4±0.3	90.9±0.3	—
MoNet	81.3±1.3	71.2±2.0	78.6±2.3	90.8±0.6	83.5±2.2	91.2±2.3	—
GS-mean	79.2±7.7	71.6±1.9	77.4±2.2	91.3±2.8	82.4±1.8	91.4±1.3	71.49±0.27
GS-maxpool	76.6±1.9	67.5±2.3	76.1±2.3	85.0±1.1	—	90.4±1.3	—
CGNN	81.4±1.6	66.9±1.8	66.6±4.4	**92.3**±0.2	80.3±2.0	91.4±1.5	58.70±2.50
GDE	78.7±2.2	71.8±1.1	73.9±3.7	91.6±0.1	82.9±0.6	92.4±2.0	56.66±10.9
BLEND	**84.8**±0.9	**75.9**±1.3	**79.5**±1.4	**92.9**±0.2	**86.9**±0.6	**92.9**±0.6	**72.56**±0.1
BLEND-kNN	**82.5**±0.9	**73.4**±0.5	**80.9**±0.7	**92.3**±0.1	**86.7**±0.6	**93.5**±0.3	—


Accuracy
90
80
70
60
50
40
30
20
Dopri5
0.25
0.5
1
2
4
8
Stepsize
Cora
Citeseer
Pubmed
CoauthorCS
Computers
Photo


BLEND
BLEND w/o positional
GAT w. POS
Accuracy
100
95
90
85
80
75
70
84.8
83.6
82.6
75.9
73.4
74.5
79.5
78.8
76.8
92.9
92.9
90.3
86.9
83.7
74
92.9
92.3
Cora
CiteSeer
PubMed
Coauthor CS
Computer
Photo
Datasets

Extensions

- Time dependent diffusivity
- Onsager diffusion

Graph Neural Diffusion provides a principled mathematical framework for
studying many popular architectures for deep learning on graphs as well as a
blueprint for developing new ones

Thank You

@b_p_chamberlain, @_JRowbottom