GRAND: Graph Neural Diffusion

Molecular Team Lecture Series
Dong-Hee Shin
05.03.22

Researchers

Benjamin Paul Chamberlain
Twitter

James Rowbottom
Twitter

Michael Bronstein
DeepMind & Twitter

Background (1)

Two perspective on Image processing:

Continuous:

“The real world is continuous as are high definition images”

“Treat an image as function, evolve it using ODEs or PDEs”

Discrete:

“A computer always model image as a tensor”

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

Background (2)

Ordinary Differential Equation (ODE):

"Differential equation containing one independent variable and the derivatives of those functions"

$$\frac{d\mathbf{z}}{dt} = f(\mathbf{z}, t), \quad f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$$

$$\frac{d\mathbf{z}}{dt} = A\mathbf{z}, \quad A \in \mathbb{R}^{n \times n}$$

$$\mathbf{z}(t) = e^{A(t-t_0)}\mathbf{z}(t_0)$$

Neural Ordinary Differential Equations

Ricky T. Q. Chen*, Yulia Rubanova*, Jesse Bettencourt*, David Duvenaud
University of Toronto, Vector Institute
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu

Abstract

We introduce a new family of deep neural network models. Instead of specifying a discrete sequence of hidden layers, we parameterize the derivative of the hidden state using a neural network. The output of the network is computed using a black-box differential equation solver. These continuous-depth models have constant memory cost, adapt their evaluation strategy to each input, and can explicitly trade numerical precision for speed. We demonstrate these properties in continuous-depth residual networks and continuous-time latent variable models. We also construct continuous normalizing flows, a generative model that can train by maximum likelihood, without partitioning or ordering the data dimensions. For training, we show how to scalably backpropagate through any ODE solver, without access to its internal operations. This allows end-to-end training of ODEs within larger models.

NIPS 2019 Best Paper

Background (3)

Partial Differential Equation (PDE):

“equation which imposes relations between the various partial derivatives of a multivariable function”

1. $u_x + u_y = 0$ (전달)
2. $u_x + yu_y = 0$ (전달)
3. $u_x + uu_y = 0$ (충격파)
4. $u_{xx} + u_{yy} = 0$ (라플라스 방정식)
5. $u_{tt} - u_{xx} + u^3 = 0$ (상호작용파)
6. $u_t + uu_x + u_{xxx} = 0$ (분산파)
7. $u_{tt} + u_{xxxx} = 0$ (진동하는 막대)
8. $u_t - iu_{xx} = 0$ $(i = \sqrt{-1})$ (양자역학)

$$u(x,\ y,\ \ldots)$$

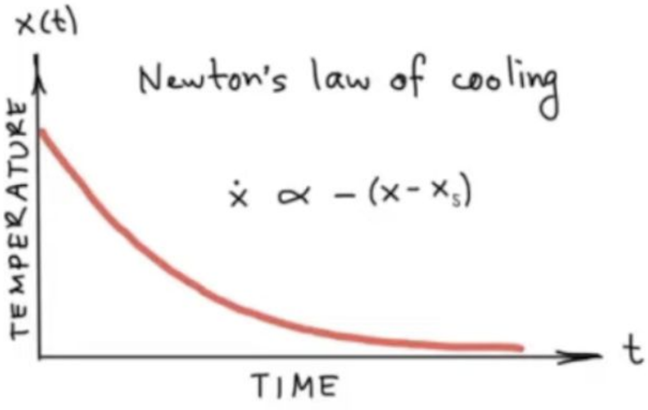
$$\frac{\partial u}{\partial x} = u_x$$

$$-\frac{\hbar^2}{2m}\nabla^2\psi + V\psi = E\psi$$

History of Diffusion (1)

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 chafed 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.*

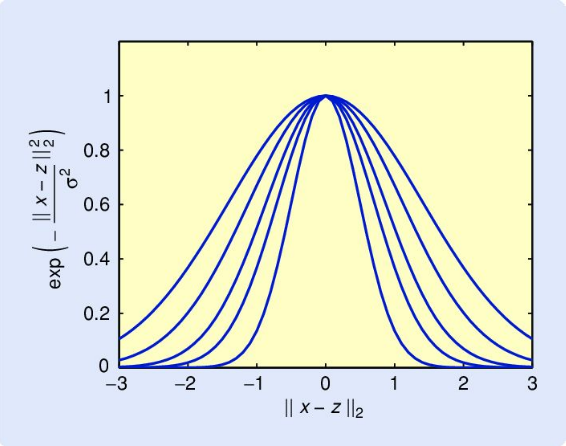
Calorum Deſcriptiones & ſigna.

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

“A Scale of the Degrees of Heat”
Newton, 1701

Heat Diffusion Equation



$$\dot{x} = \mathrm{a}\Delta x.$$

This PDE is linear and its solution can be given in closed form as the convolution of the initial temperature distribution with a time-dependent Gaussian kernel

$$x(u,t) = x(u,0) * \exp(-|u|^2/4t).$$

$$\dot{x}(u,t) = \operatorname{div}(a(u,t)\nabla x(u,t))$$

History of Diffusion (2-1)

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 ”

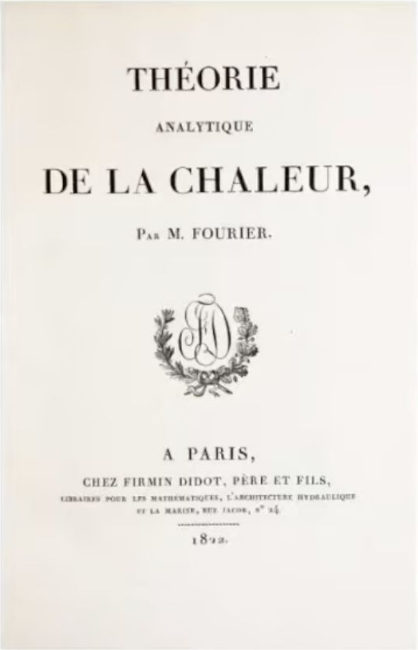
$$\vec{q} = -k\nabla T$$

where

q is the vector of local heat flux density [$W.m^{-2}$]

k is the materials conductivity [$W.m^{-1}.K^{-1}$]

∇T is the temperature gradient [$K.m^{-1}$]



THÉORIE

ANALYTIQUE

DE LA CHALEUR,

PAR M. FOURIER.

A PARIS,

CHEZ FIRMIN DIDOT, PÈRE ET FILS,

LIBRAIRES POUR LES MATHÉMATIQUES, L'ARCHITECTURE HYDRAULIQUE

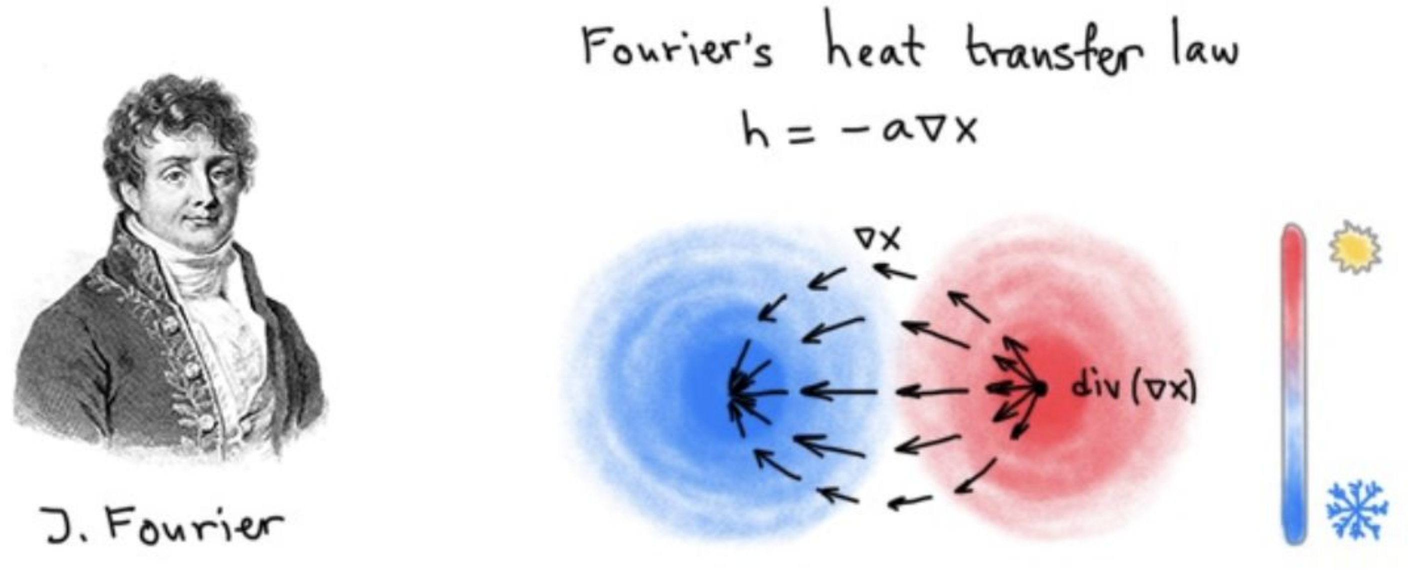
ET LA MARINE, RUE JACOB, n° 24.

1822.

Joseph Fourier

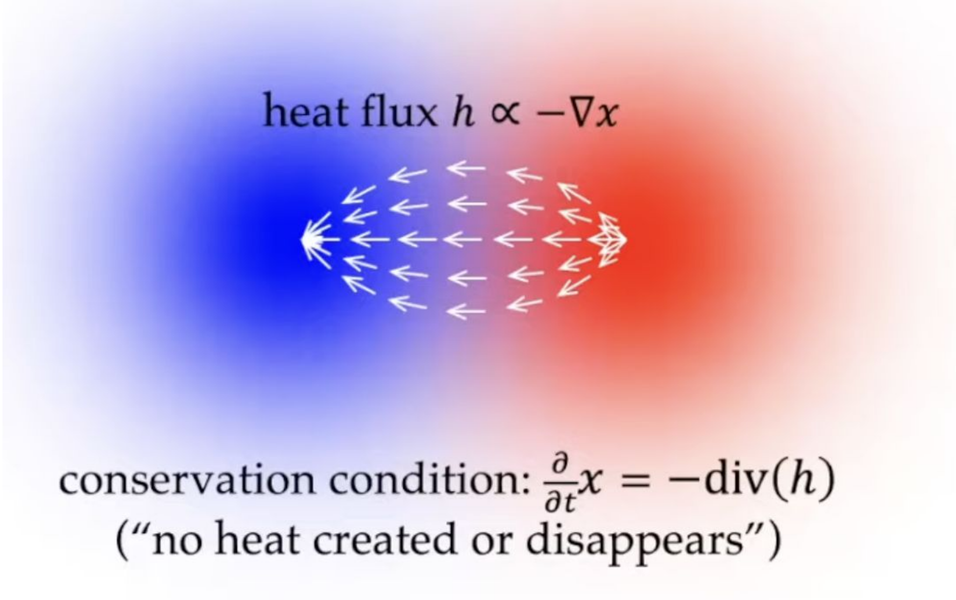
Fourier
1822

History of Diffusion (2-2)



Fourier
1822

History of Diffusion (3)



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

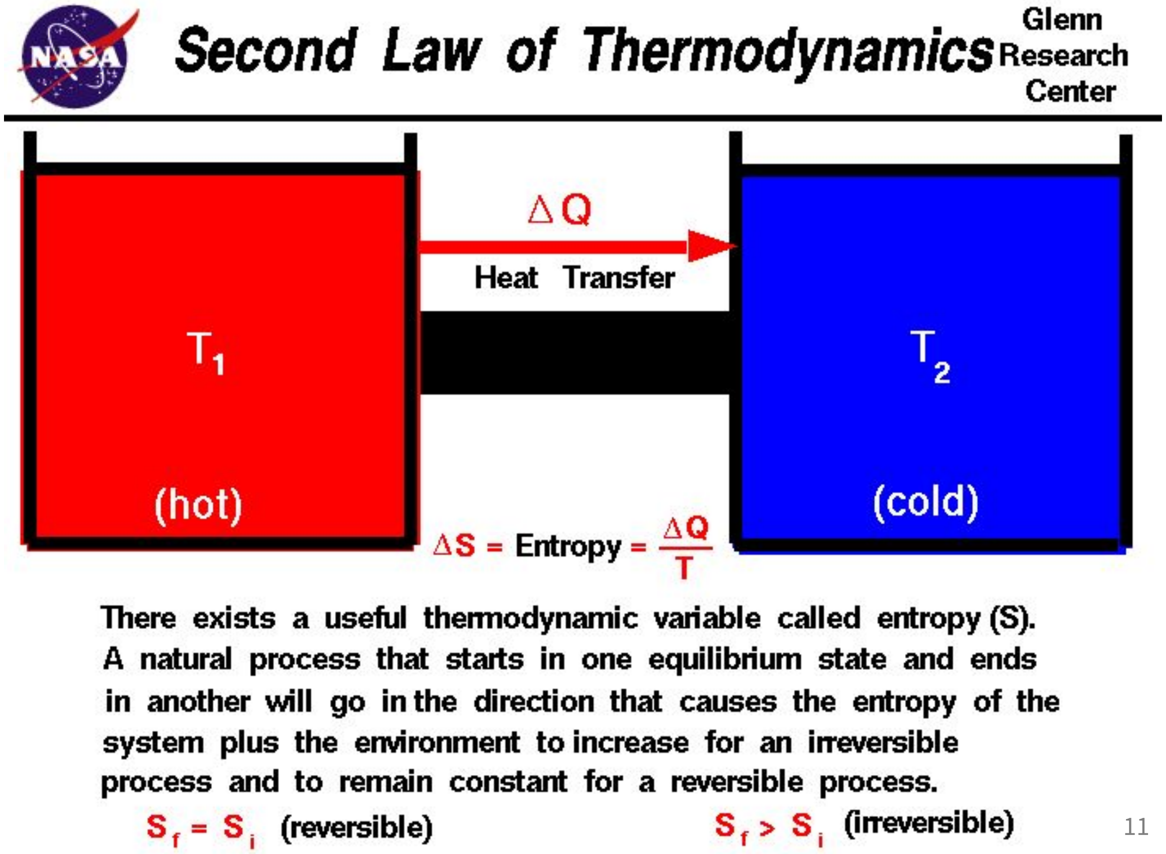
Adolf Eugen Fick

Fick

1855

Diffusion and Entropy

Diffusion is a direct result of the second law or entropy:



History of Diffusion (4)

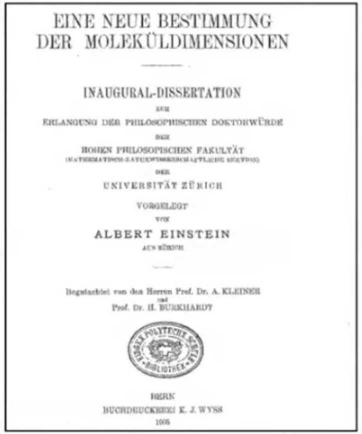
Brownian Motion :

“Random motion of particles suspended in a medium (a liquid or a gas). ”

“Albert Einstein published a paper where he modeled the motion of the pollen particles as being moved by individual water molecules”

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 PHILOSOPISCHEN FAKULTÄT
DER
UNIVERSITÄT ZÜRICH
VORGELEGT
VON
ALBERT EINSTEIN

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

BERN
BUCHDRUCKEREI K. J. WYSS

Albert Einstein

Einstein
1905

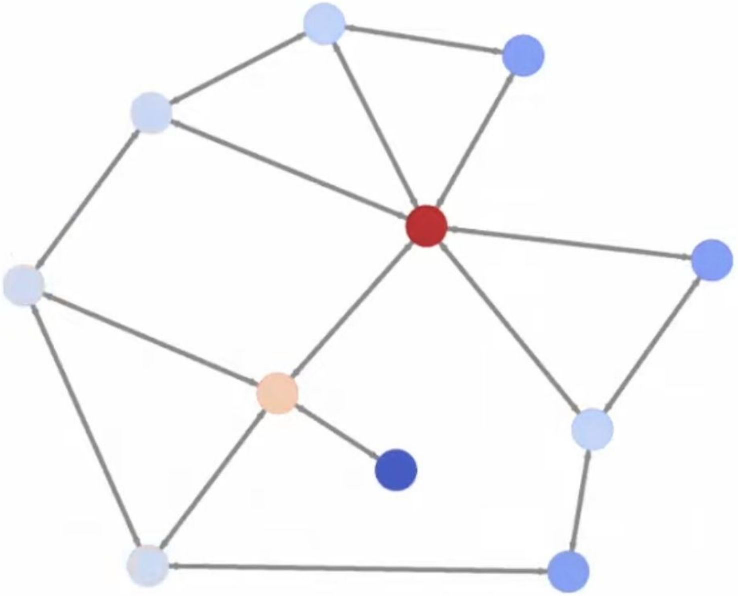
History of Diffusion (5)

Diffusion in a Discrete Domain:

“Google Page Rank”

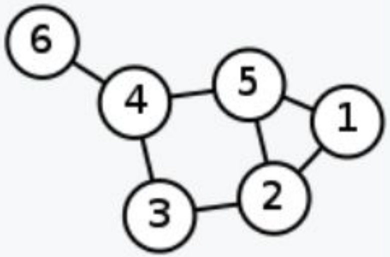
“Laplacian Matrix”

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



Pierre-Simon Laplace

Laplacian Matrix

Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

The symmetric normalized Laplacian matrix is defined as:[1]

$$L^{\text{sym}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}},$$

The elements of L^{sym} are given by

$$L^{\text{sym}}_{i,j} := \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0 \\ -\dfrac{1}{\sqrt{\deg(v_i)\deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

Diffusion Equation

Homogeneous: diffusion is same everywhere and in every direction

Non-Homogeneous: diffusion is expressed through diffusivity constant

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

Homogeneous
Isotropic

Position-dependent diffusivity

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

Non-homogeneous
Isotropic

Position & direction dependent diffusivity

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

Non-homogeneous
Anisotropic

Diffusion in Image Processing

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

Homogeneous diffusion

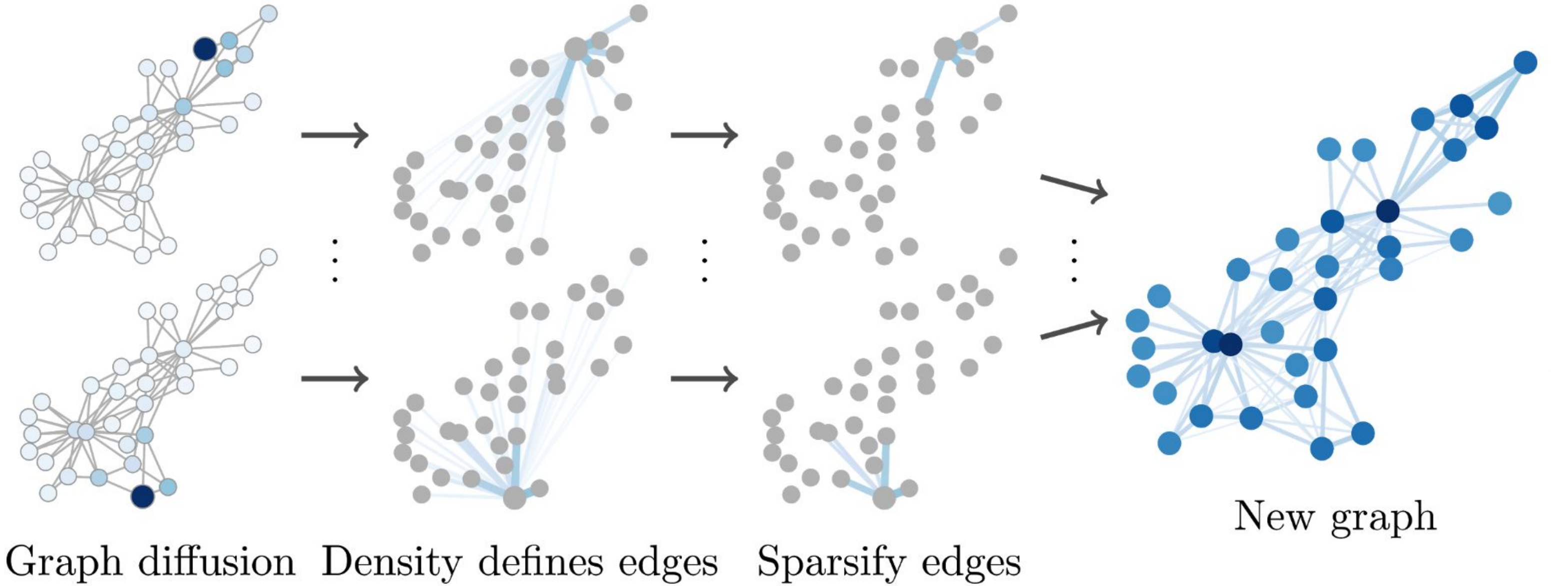
Non-homogeneous diffusion

Connecting PDEs to GNNs

MPN is Diffusion

Message passing is just a diffusion process:

"In short, message passing is nothing but discrete diffusion"



Spatial Discretization

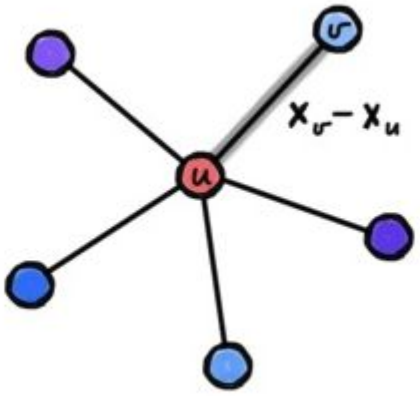
$$\frac{\partial x(u,t)}{\partial t} = \mathrm{div}[g(u, x(u,t), t)\nabla x(u,t)],$$

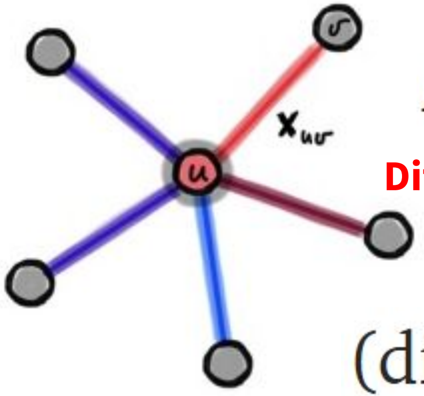
GNNs exchange information between adjacent nodes = Diffusion

Spatial Derivatives = Difference between adjacent node features

$$\dot{\mathbf{X}}(t) = \mathrm{div}(\mathbf{A}(\mathbf{X}(t))\nabla\mathbf{X}(t))$$

$$(\nabla\mathbf{X})_{uv} = \mathbf{x}_v - \mathbf{x}_u$$





Gradient → flow along edges

$$\mathbf{A}(\mathbf{X}) = \mathrm{diag}(a(\mathbf{x}_u, \mathbf{x}_v))$$

Diffusivity → strength of diffusion

$$(\mathrm{div}(\mathbf{X}))_u = \textstyle\sum_v w_{uv}\, \mathbf{x}_{uv}$$

Divergence → Agg of edges

Graph Diffusion Equation (1)

$$\dot{\mathbf{X}}(t) = \mathrm{div}(\mathbf{A}(\mathbf{X}(t))\nabla \mathbf{X}(t))$$

Matrix Differential Equation

$$\dot{\mathbf{X}}(t) = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t).$$

Temporal Discretization

$$[\mathbf{X}(k+1) - \mathbf{X}(k)]/\tau = [\mathbf{A}(\mathbf{X}(k)) - \mathbf{I}]\mathbf{X}(k)$$

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

$$\mathbf{X}(k+1) = [(1-\tau)\mathbf{I} + \tau\mathbf{A}(\mathbf{X}(k))]\mathbf{X}(k) = \mathbf{Q}(k)\mathbf{X}(k)$$

Explicit/Forward Euler Scheme

➢ **Recursive form with linear operator Q(k)**

Graph Diffusion Equation (2)

$$\dot{\mathbf{X}}(t) = (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t).$$

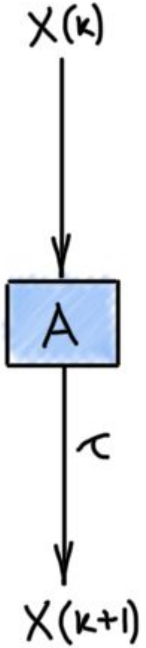
$$\mathbf{X}(k+1) = [(1-\tau)\mathbf{I} + \tau\mathbf{A}(\mathbf{X}(k))]\mathbf{X}(k) = \mathbf{Q}(k)\mathbf{X}(k)$$

$$[(1+\tau)\mathbf{I} - \tau\mathbf{A}(\mathbf{X}(k))]\mathbf{X}(k+1) = \mathbf{B}(k)\mathbf{X}(k+1) = \mathbf{X}(k)$$

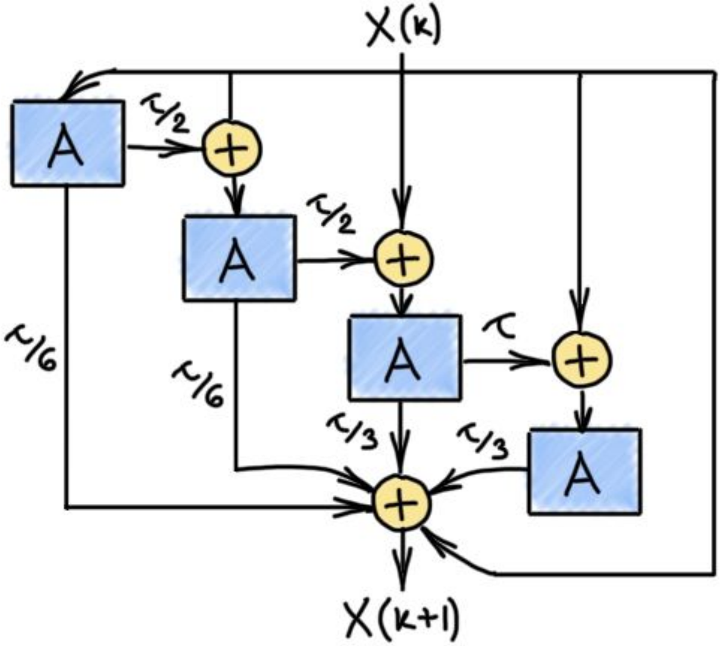
Semi/Implicit Scheme

- **Need to solve a linear system (Inversion of B)**

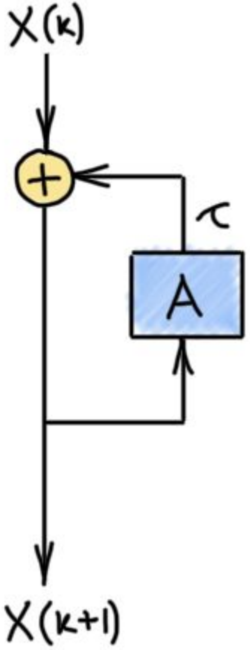
Different Discretization Scheme



EXPLICIT



RUNGE-KUTTA 4



IMPLICIT

GRAND

$$\dot{\mathrm{X}}(t) = \mathrm{div}(\mathrm{A}(\mathrm{X}(t))\nabla\mathrm{X}(t)) \qquad \mathrm{A}(\mathrm{X}) = \mathrm{diag}(a(\mathrm{x}_u, \mathrm{x}_v))$$

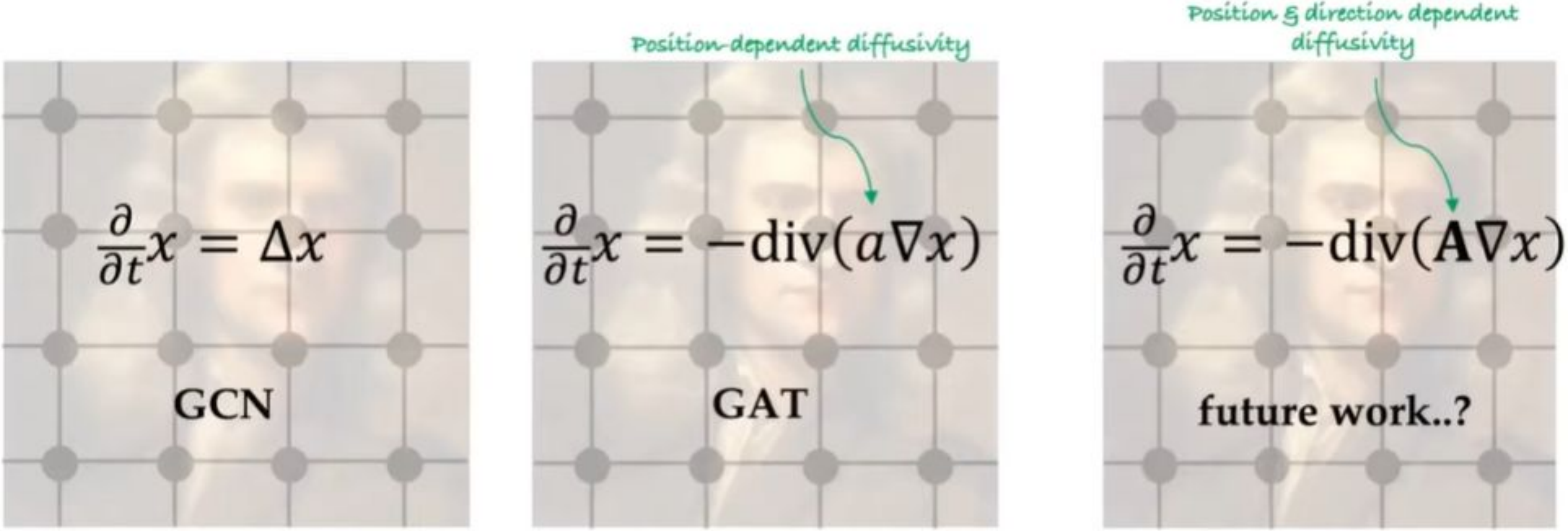
Recall that a is function that determine similarity between noc
i* and node *j

$$\frac{\partial \mathbf{x}(t)}{\partial t} = \mathrm{div}[\mathbf{G}(\mathbf{x}(t), t)\nabla\mathbf{x}(t)] \qquad \mathbf{G} = \mathrm{diag}(a(x_i(t), x_j(t), t))$$

Almost Every GNN architecture can be formalized as a discretization scheme!! (GAT = Explicit Scheme → Attention = Diffusivity)

1) The discrete time index = layer of GNN
2) Running the diffusion for multiple iterations = Applying GNN layer multiple times
3) Output = the solution of X(t) in diffusion equation at some end time *T*

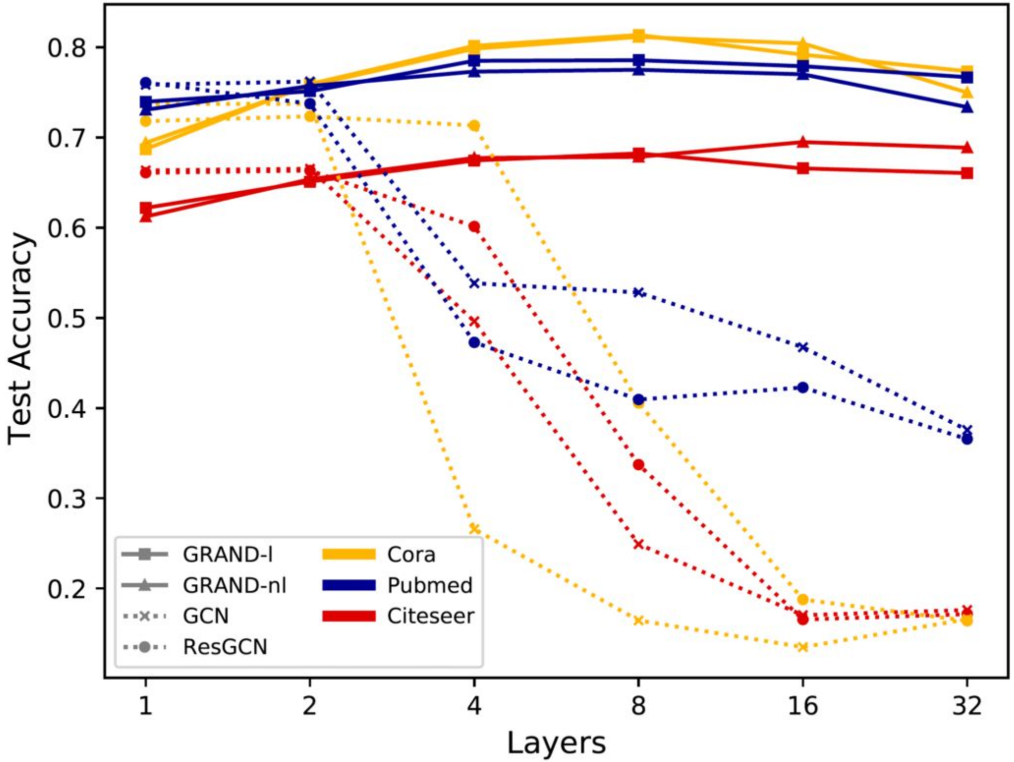
Diffusivity for GNNs



Result (1)

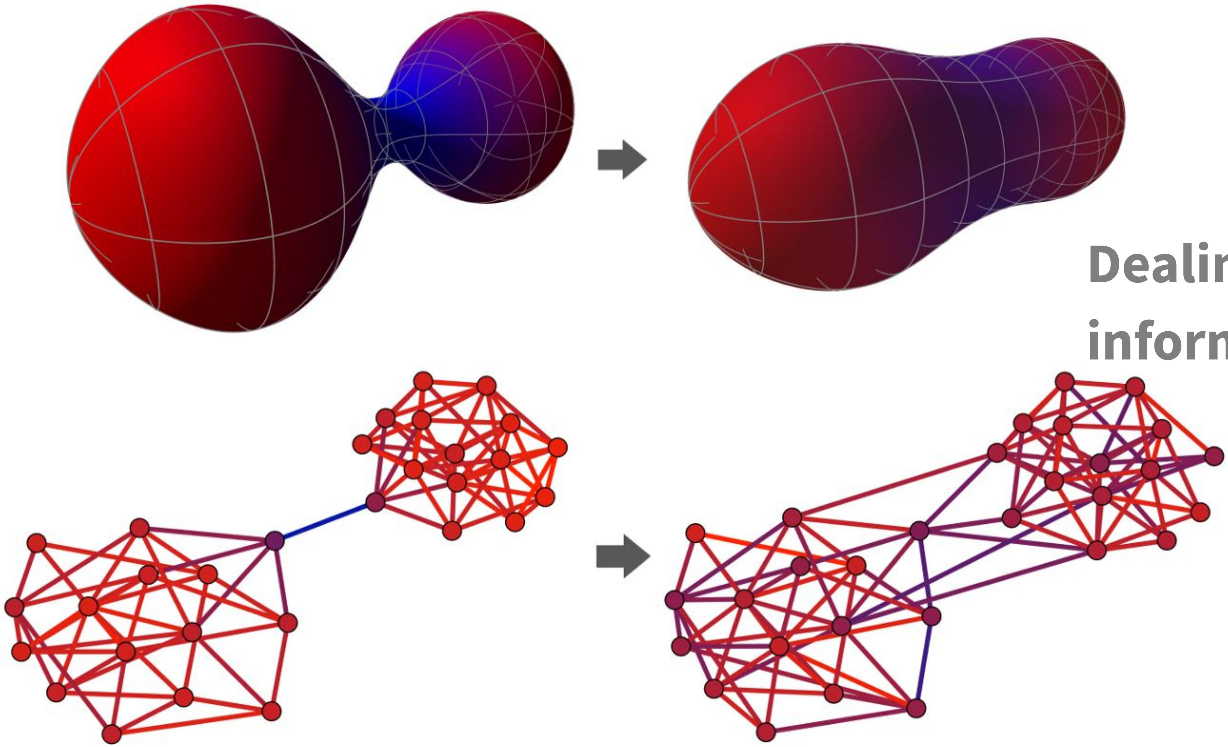
Planetoid splits	CORA	CiteSeer	PubMed
GCN	81.9 ± 0.8	69.5 ± 0.9	79.0 ± 0.5
GAT	82.8 ± 0.5	71.0 ± 0.6	77.0 ± 1.3
MoNet	82.2 ± 0.7	70.0 ± 0.6	77.7 ± 0.6
GS-maxpool	77.4 ± 1.0	67.0 ± 1.0	76.6 ± 0.8
Lanczos	79.5 ± 1.8	66.2 ± 1.9	78.3 ± 0.3
AdaLanczos	80.4 ± 1.1	68.7 ± 1.0	78.1 ± 0.4
CGNN†	81.7 ± 0.7	68.1 ± 1.2	$\mathbf{80.2 \pm 0.3}$
GDE*	$\mathbf{83.8 \pm 0.5}$	$\mathbf{72.5 \pm 0.5}$	79.9 ± 0.3
GODE*	83.3 ± 0.3	72.4 ± 0.6	80.1 ± 0.3
GRAND-l (ours)	$\mathbf{84.7 \pm 0.6}$	$\mathbf{73.3 \pm 0.4}$	$\mathbf{80.4 \pm 0.4}$
GRAND-nl (ours)	$\mathbf{83.6 \pm 0.5}$	70.8 ± 1.1	79.7 ± 0.3
GRAND-nl-rw (ours)	82.9 ± 0.7	$\mathbf{73.6 \pm 0.3}$	$\mathbf{81.0 \pm 0.4}$

Result (2)



1) **Explicit GNN layers are replaced with the continuous analog with diffusion time**
2) **Diffusion Model can solve the oversmoothing problem**

Graph Rewiring



Dealing with Scalability and information bottleneck

Summary

1) **PDEs are intimately related to GNNs (GNN is nothing but numerical solution to PDE)**
2) **New perspective on old problem like oversmoothing**
3) **New architecture inspired by numerical ODE solvers**
4) **Stability conditions**
5) **Access to new domains inspired by PDEs**

Thank you

Dong-Hee Shin
(dongheeshin@korea.ac.kr)