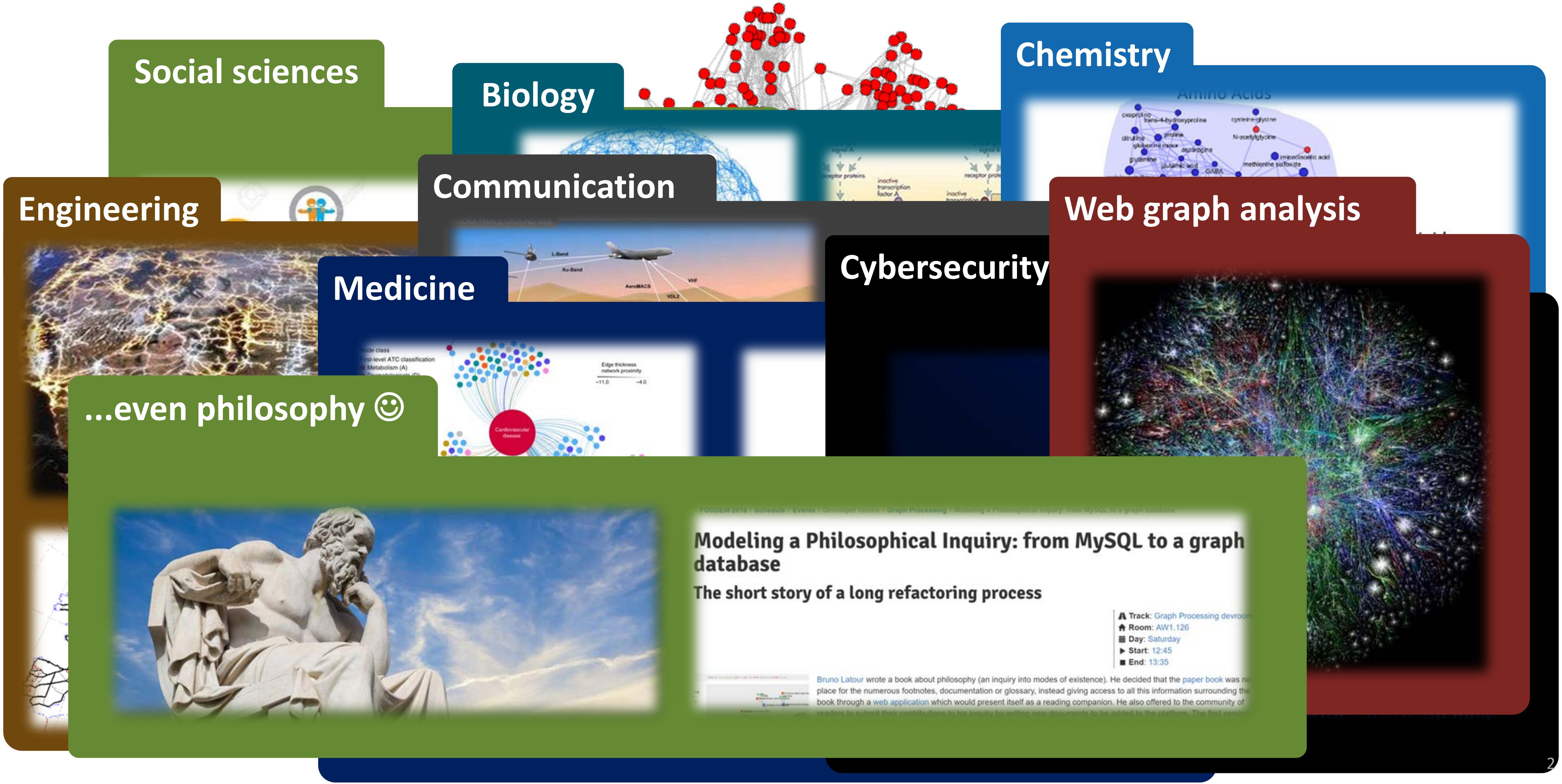
M. Besta, P. Renc, R. Gerstenberger, P. S. Labini, A. Ziogas, T. Chen, L. Gianinazzi, F. Scheidl, K. Szenes, A. Cariget, P. Iff, G. Kwasniewski, R. Kanakagiri, C. Ge, S. Jaeger, J. Was, F. Vella, T. Hoefler

Artifacts Available V1.1 · Artifacts Evaluated Functional V1.1 · Results Reproduced V1.1

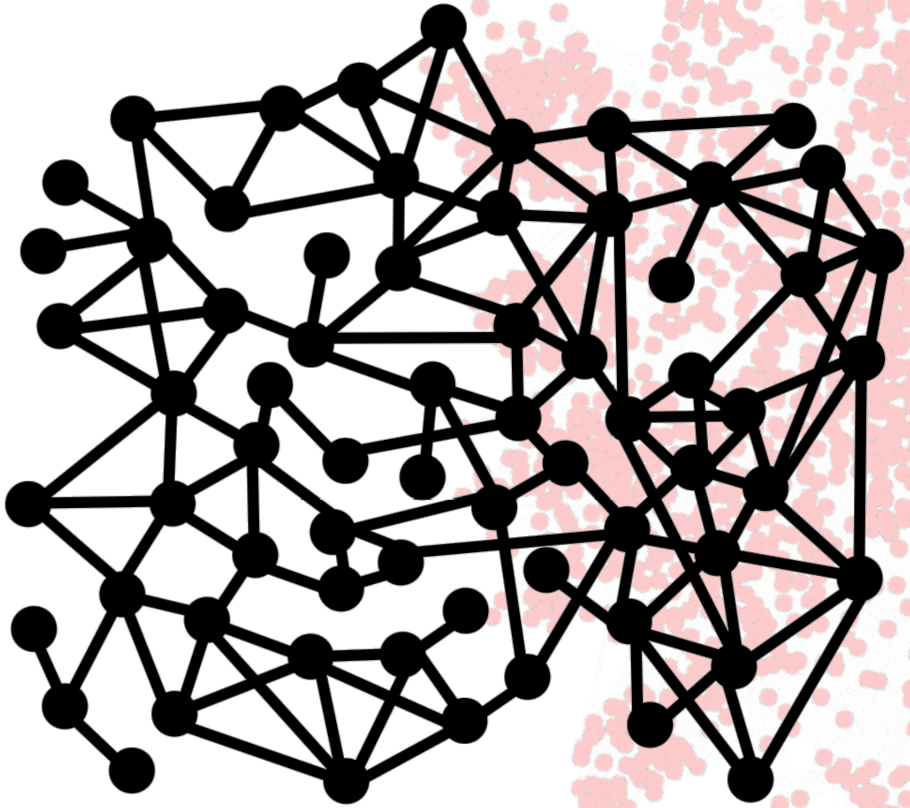
High-Performance and Programmable Attentional Graph Neural Networks with Global Tensor Formulations

Graphs are Powerful and Ubiquitous!

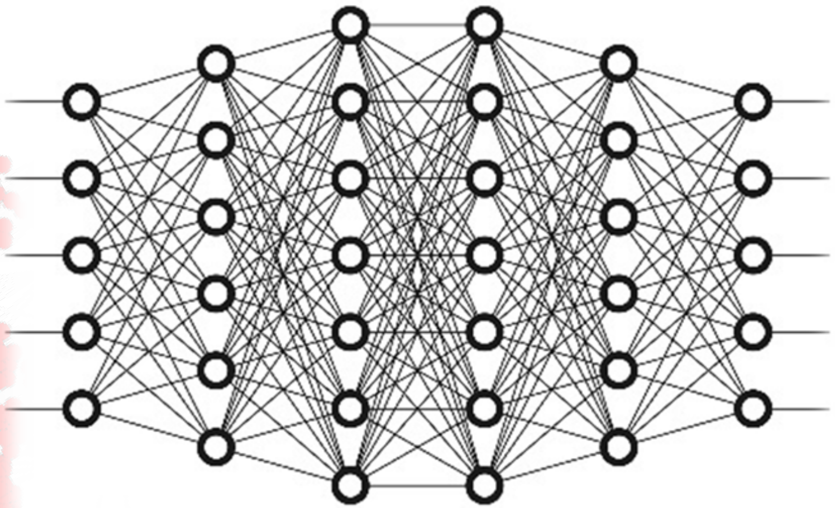


Graphs + Deep Learning = Graphs Neural Networks (GNNs)

In the last 5 years, learning on graphs exploded



+



Let's See Some Recent Success Stories of GNNs

Article

A graph placement methodology for fast chip design

https://doi.org/10.1038/s41586-021-03
Received: 3 November 2020
Accepted: 13 April 2021
Published online: 9 June 2021
Check for updates

Article

Advancing mathematics by guiding human intuition with AI

https://doi.org/10.1038/s41586-021-04086-x
Received: 10 July 2021
Accepted: 30 September 2021
Published online: 1 December 2021
Open access

Alex Da
Nenad
Marc La

The pra
formul

Article

Highly accurate protein structure prediction with AlphaFold

https://doi.org/10.1038/s41586-021-03819-2
Received: 11 May 2021
Accepted: 12 July 2021
Published online: 15 July 2021
Open access
Check for updates

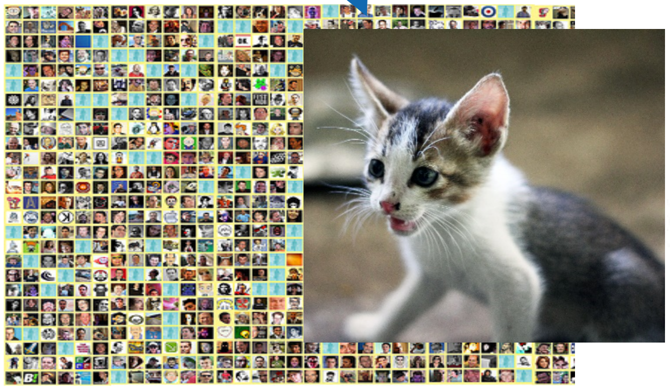
John Jumper[1,4✉], Richard Evans[1,4], Alexander Pritzel[1,4], Tim Green[1,4], Michael Figurnov[1,4], Olaf Ronneberger[1,4], Kathryn Tunyasuvunakool[1,4], Russ Bates[1,4], Augustin Žídek[1,4], Anna Potapenko[1,4], Alex Bridgland[1,4], Clemens Meyer[1,4], Simon A. A. Kohl[1,4], Andrew J. Ballard[1,4], Andrew Cowie[1,4], Bernardino Romera-Paredes[1,4], Stanislav Nikolov[1,4], Rishub Jain[1,4], Jonas Adler[1], Trevor Back[1], Stig Petersen[1], David Reiman[1], Ellen Clancy[1], Michal Zielinski[1], Martin Steinegger[2,3], Michalina Pacholska[1], Tamas Berghammer[1], Sebastian Bodenstein[1], David Silver[1], Oriol Vinyals[1], Andrew W. Senior[1], Koray Kavukcuoglu[1], Pushmeet Kohli[1] & Demis Hassabis[1,4✉]

Proteins are essential to life, and understanding their structure can facilitate a

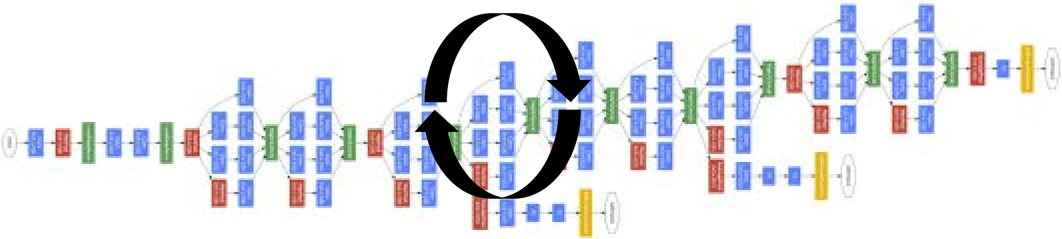
Deep Learning (DL) in a Nutshell

Samples

Learning a model



f(x)



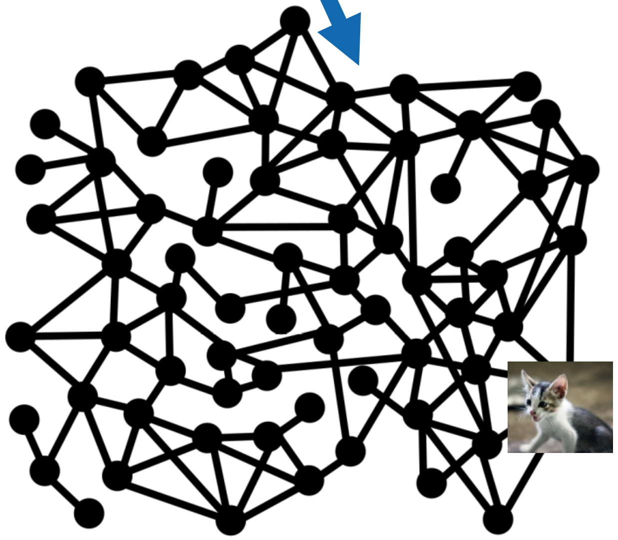
layer-wise weight update

Cat	0.54	1.00
Dog	0.28	0.00
Airplane	0.07	0.00
Horse	0.04	0.00
Bicycle	0.33	0.00
Truck	0.02	0.00
	0.02	0.00

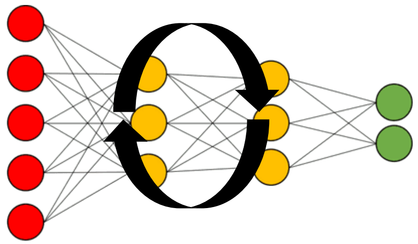
A Primer on Graph Neural Networks (GNNs)

Samples

These could still be photos, but now forming **explicit relations**, e.g., two photos are related if they were taken at the same place.



f(x)



layer-wise weight update

Cat	0.54
Dog	0.28
Airplane	0.07
Horse	0.04
Bicycle	0.33
Truck	0.02
	0.02

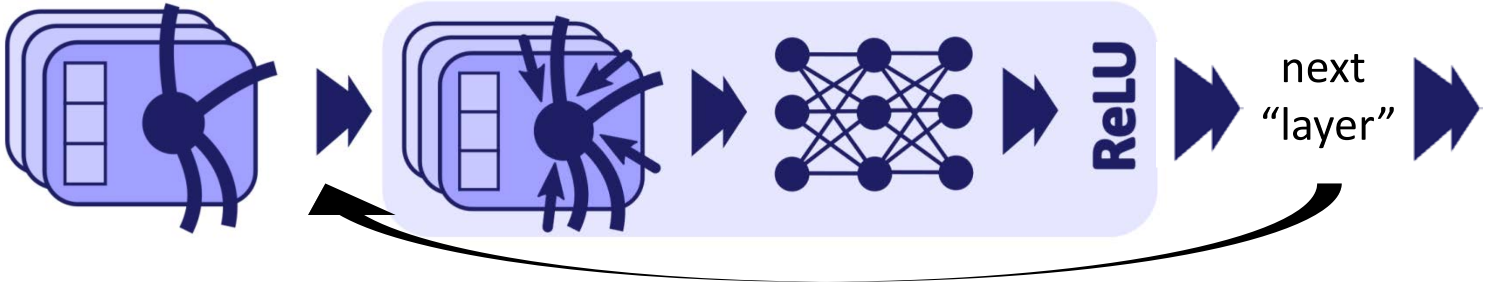
Cat	1.00
Dog	0.00
Airplane	0.00
Horse	0.00
Bicycle	0.00
Truck	0.00
	0.00

input samples

gather from neighbors (sparse)

apply DNN (often simple)

optional nonlinearity



output vector (can be used for specific tasks)

Taxonomy of Mathematical Formulations of GNNs

Local GNN formulations

Formulations based on functions operating on single vertices & edges

Vertex feature vector

GNN layer

Functions that specify the model details

$$\mathbf{h}_i^{(l+1)} = \boxed{\phi}\left(\mathbf{h}_i^{(l)}, \boxed{\bigoplus_{j \in N(i)}} \boxed{\psi}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

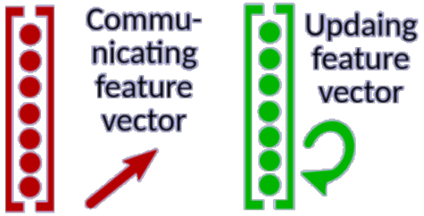
Vertex ID

Neighbors

Taxonomy of Mathematical Formulations of GNNs

Red color: communication

Green color: computation

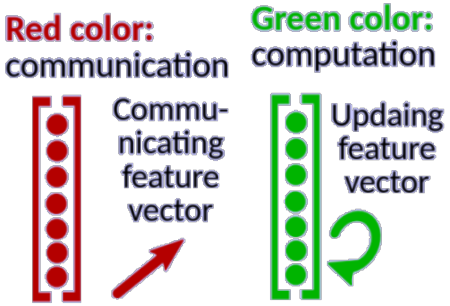


Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

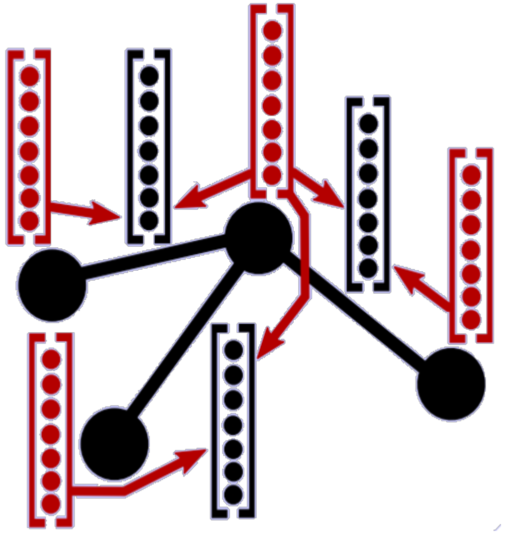
Taxonomy of Mathematical Formulations of GNNs



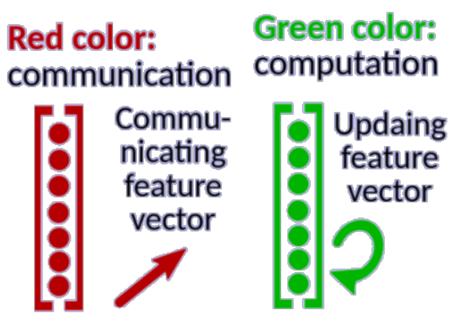
Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \boxed{\psi}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$



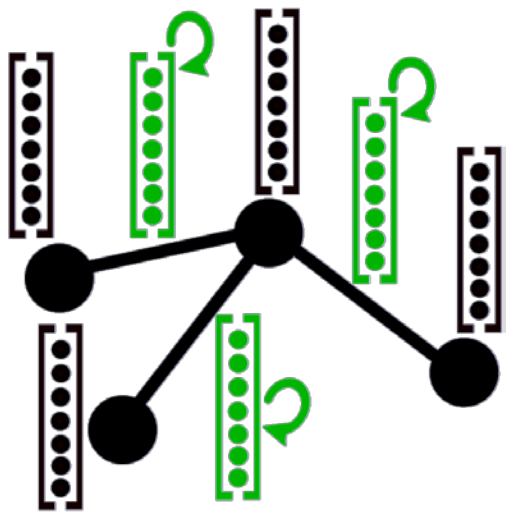
Taxonomy of Mathematical Formulations of GNNs



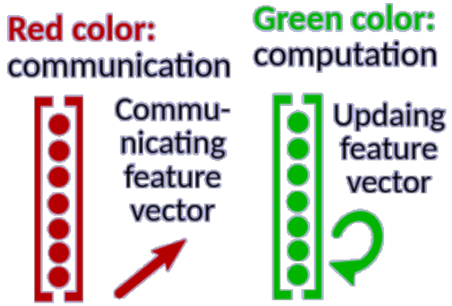
Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \boxed{\psi}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$



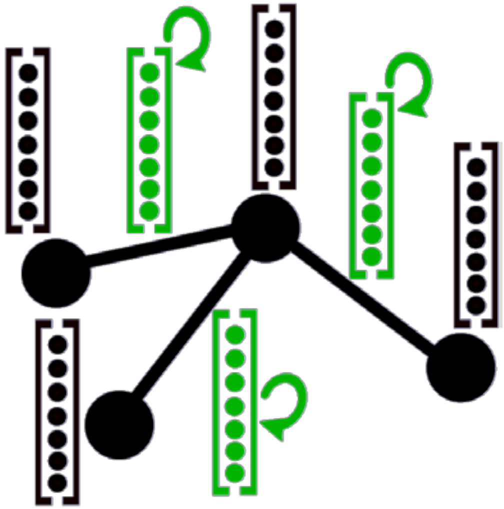
Taxonomy of Mathematical Formulations of GNNs



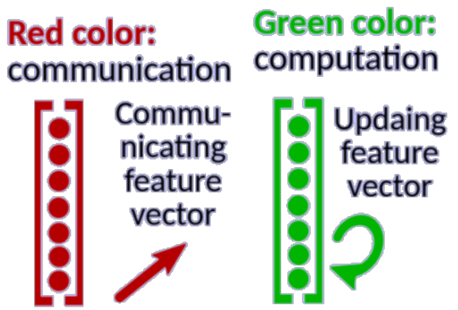
Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \boxed{\bigoplus}_{j \in N(i)} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$



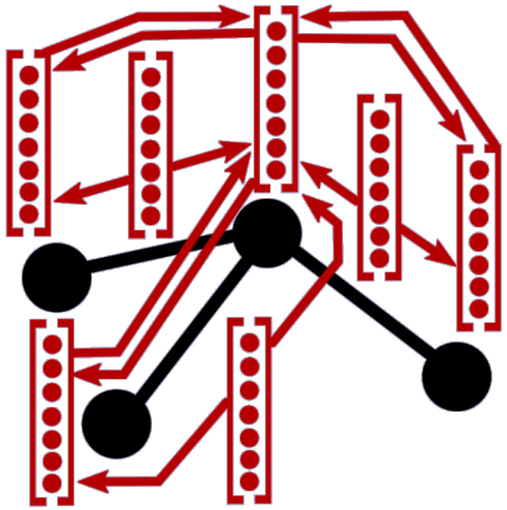
Taxonomy of Mathematical Formulations of GNNs



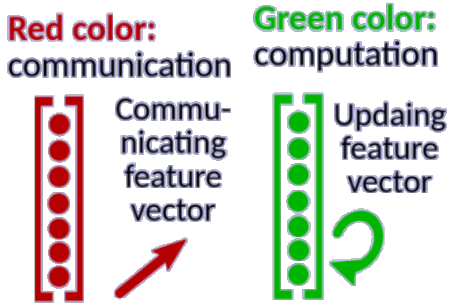
Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \boxed{\bigoplus_{j \in N(i)}} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$



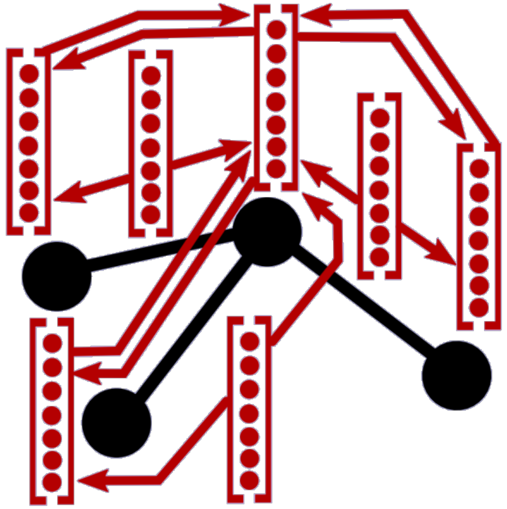
Taxonomy of Mathematical Formulations of GNNs



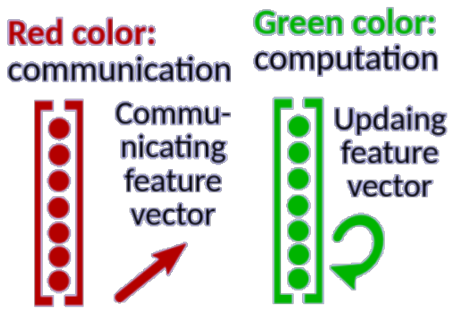
Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\mathbf{h}_i^{(l+1)} = \boxed{\phi}\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$



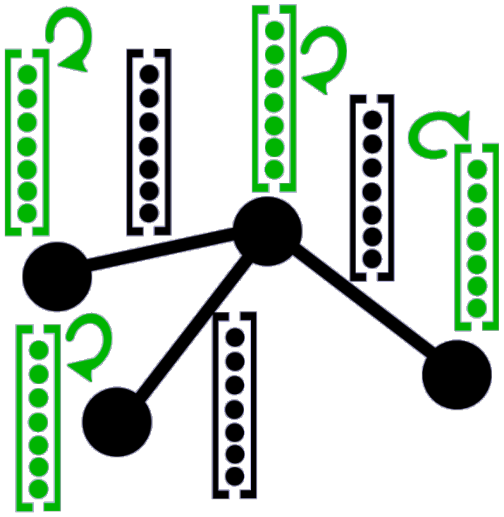
Taxonomy of Mathematical Formulations of GNNs



Local GNN formulations

Formulations based on functions operating on single vertices & edges

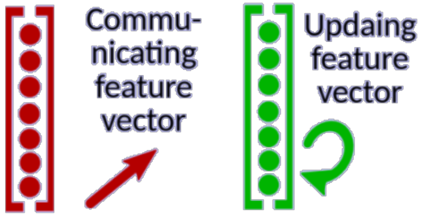
$$\mathbf{h}_i^{(l+1)} = \boxed{\phi}\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$



Taxonomy of Mathematical Formulations of GNNs

Red color: communication

Green color: computation



Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

Taxonomy of Mathematical Formulations of GNNs

Local GNN formulations

Formulations based on functions operating on single vertices & edges

$$\cdots$$

$$\mathbf{h}_{i-1}^{(l+1)} = \phi\left(\mathbf{h}_{i-1}^{(l+1)} \bigoplus_{j \in N(i-1)} \psi\left(\mathbf{h}_{i-1}^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

$$\mathbf{h}_{i}^{(l+1)} = \phi\left(\mathbf{h}_{i}^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_{i}^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

$$\mathbf{h}_{i+1}^{(l+1)} = \phi\left(\mathbf{h}_{i+1}^{(l+1)} \bigoplus_{j \in N(i+1)} \psi\left(\mathbf{h}_{i+1}^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

$$\cdots$$

Taxonomy of Mathematical Formulations of GNNs

Local GNN formulations

Formulations based on functions operating on single vertices & edges

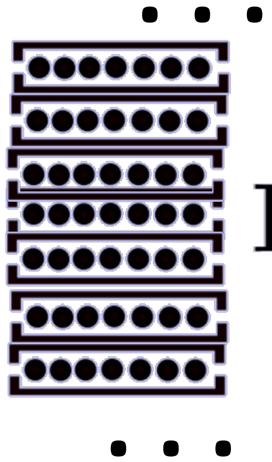
$$\mathbf{h}_{i-1}^{(l+1)} = \phi\left(\mathbf{h}_{i-1}^{(l+1)} \bigoplus_{j \in N(i-1)} \psi\left(\mathbf{h}_{i-1}^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

$$\mathbf{h}_{i}^{(l+1)} = \phi\left(\mathbf{h}_{i}^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_{i}^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

$$\mathbf{h}_{i+1}^{(l+1)} = \phi\left(\mathbf{h}_{i+1}^{(l+1)} \bigoplus_{j \in N(i+1)} \psi\left(\mathbf{h}_{i+1}^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

Global GNN formulations

Formulations based on operations on matrices grouping all vertex & edge related vectors



All vertex feature vectors grouped together

Model parameters

$$\mathbf{H}^{l+1} = \sigma\left(\mathbf{Z}\right), \quad \mathbf{Z} = \boxed{\Psi}\mathbf{H}\mathbf{W}$$

Model details (a transformation of, among others, the adjacency matrix)

Taxonomy of Mathematical Formulations of GNNs

Local GNN formulations

Formulations based on functions operating on single vertices & edges

“Per-vertex” formulations can’t expose data reuse!

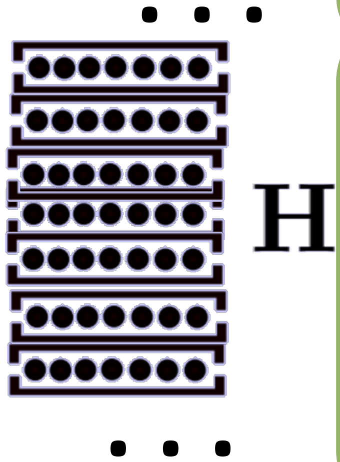
$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

$$\mathbf{h}_{i+1}^{(l+1)} = \phi\left(\mathbf{h}_{i+1}^{(l+1)} \bigoplus_{j \in N(i+1)} \psi\left(\mathbf{h}_{i+1}^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

Global GNN formulations

Formulations based on operations on matrices grouping all vertex & edge related vectors

All



Global formulations can utilize optimal linear algebra algorithms!

- Communication-avoiding 2.5D MMM
- Tiling
- Kernel fusion
- …

Taxonomy of Mathematical Formulations of GNNs

Problem:
Finding global formulations may be challenging

Global formulations are known for simple models such as Convolutional GNNs

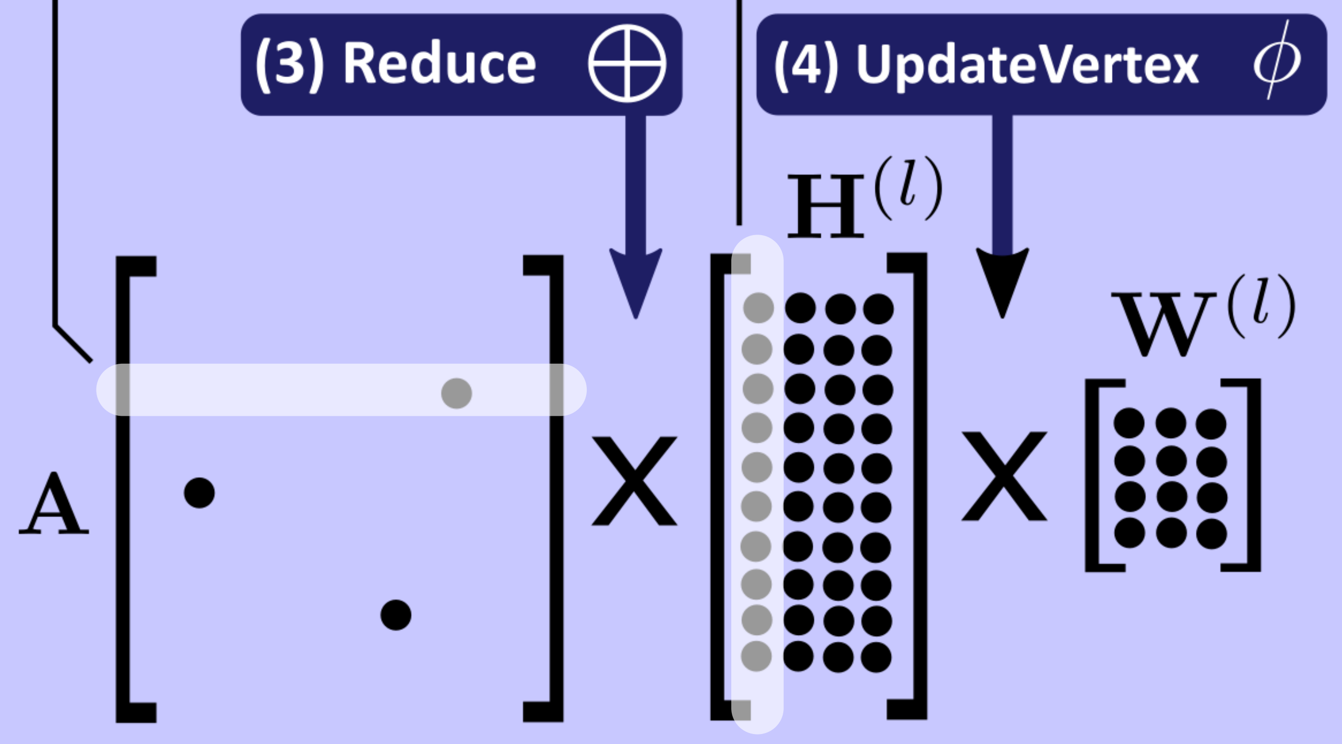
Global Formulations of GNN Models

$$\mathbf{H}^{(l+1)} = \mathbf{A} \times \mathbf{H}^{(l)} \times \mathbf{W}^{(l)}$$

The simplest model: Graph Convolution Network [1]

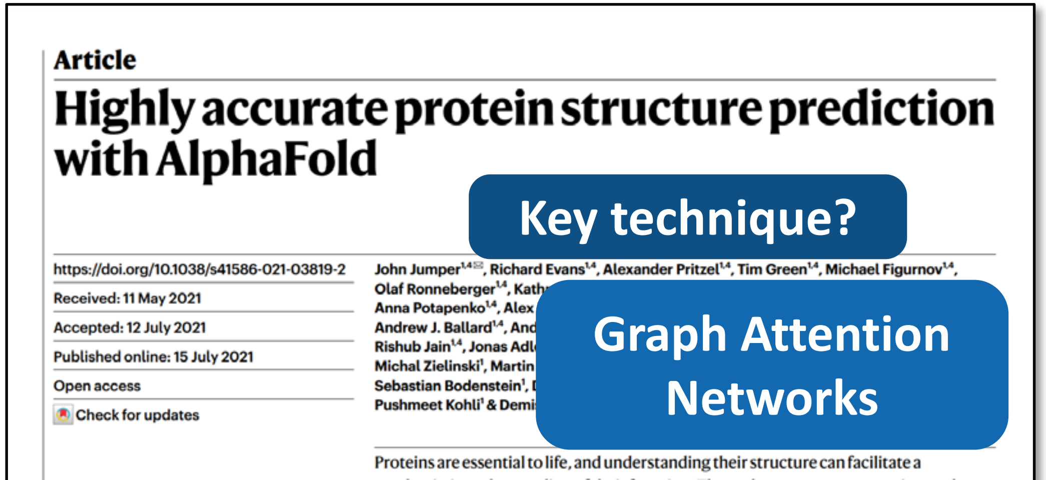
Highlighted row corresponds to the neighbors of a specific vertex *v*, whose feature vector is being computed

Highlighted column corresponds to the specific feature *f* that is being computed for vertex *v*



What are the global formulations of more complex models, such as attentional GNNs?

Also, why do we care?


Article

Highly accurate protein structure prediction with AlphaFold

https://doi.org/10.1038/s41586-021-03819-2
Received: 11 May 2021
Accepted: 12 July 2021
Published online: 15 July 2021
Open access
Check for updates

John Jumper[1,4], Richard Evans[1,4], Alexander Pritzel[1,4], Tim Green[1,4], Michael Figurnov[1,4], Olaf Ronneberger[1,4], Kath… Anna Potapenko[1,4], Alex… Andrew J. Ballard[1,4], And… Rishub Jain[1,4], Jonas Adl… Michal Zielinski[1], Martin… Sebastian Bodenstein[1], … Pushmeet Kohli[1] & Demi…

Proteins are essential to life, and understanding their structure can facilitate a

Key technique?

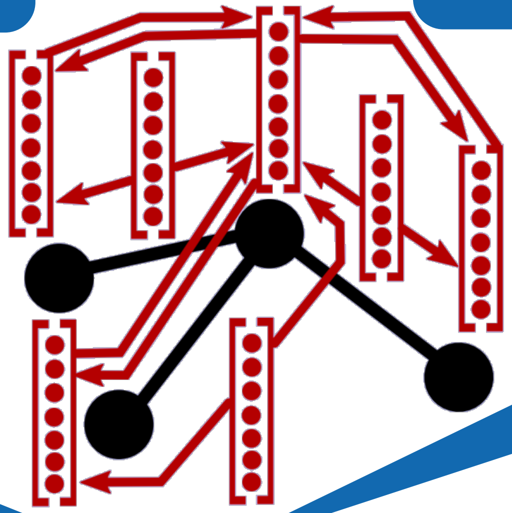
Graph Attention Networks

[1] T. Kipf et al. Semi-Supervised Classification with Graph Convolutional Networks. ICLR. 2017.

Attention in GNN Models

Convolutional GNN

Attentional GNN

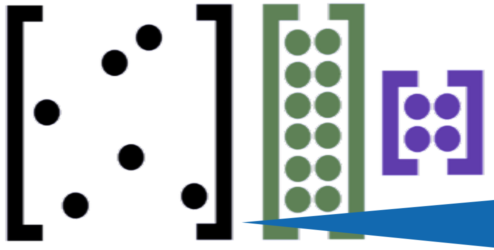


The contribution of neighbors is **fixed**
e.g., sum

The contribution of neighbors is **learnable**

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in N(i)} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right)$$

$$\mathbf{H}^{l+1} =$$



Static, binary matrix
adjacency matrix of the graph

Matrix Ψ with dynamic attention scores

Attention in GNN Models

Convolutional GNN

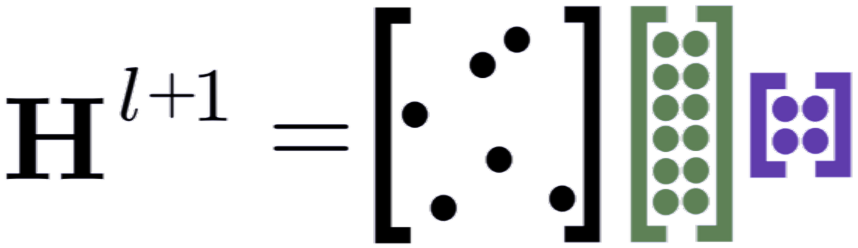
Attentional GNN

We provide *generic* global formulations for any attentional GNNs, both for the **forward** and the **backward** propagation pass

Static, binary matrix *adjacency matrix of the graph*

Matrix Ψ with dynamic attention scores

Attention in GNN Models – Forward Pass



Formulating ψ is the „crux" of devising a concrete formulation for a specific model

$$\mathbf{H}^{l+1} = \sigma\left(\mathbf{Z}\right), \quad \mathbf{Z} = \Psi \mathbf{H} \mathbf{W}$$

weights

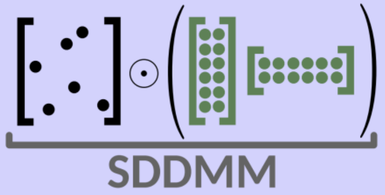
Non-linearity

A sparse *n x n* tensor with attention scores, **model specific**

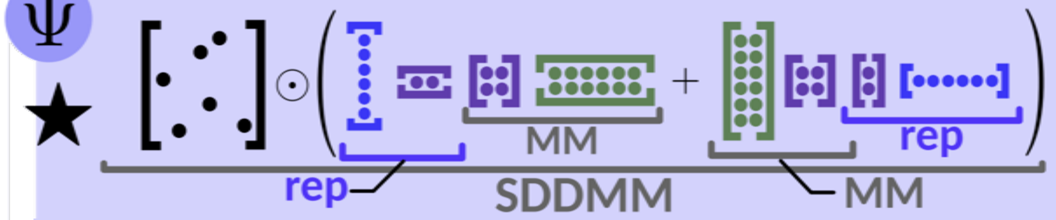
Features from previous layer

Vanilla Attention

$\Psi = \mathcal{A} \odot \mathbf{H}_\times$



Graph Attention Network (GAT)

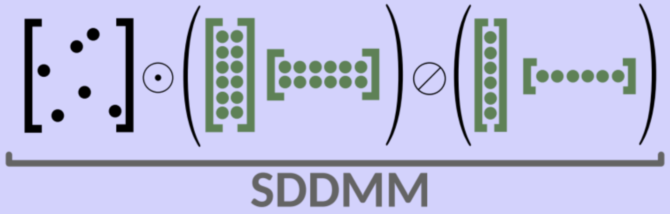


$$\Psi = \mathrm{sm}(\mathcal{T}) \quad \mathcal{T} = \mathcal{A} \odot \exp\left(\sigma\left(\mathbf{C}\right)\right)$$

$$\mathbf{C} = \mathrm{rep}_n^T((\mathbf{H}'\underline{\mathbf{a}})^T) + \mathrm{rep}_n(\mathbf{H}'\overline{\mathbf{a}})$$

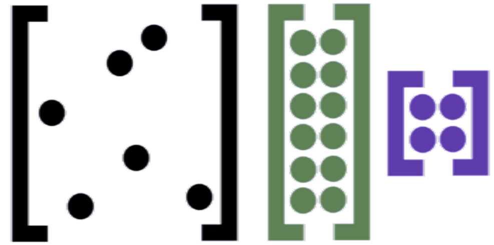
Attention-based GNN (AGNN)

$\Psi = \mathcal{A} \odot \mathbf{H}_\times \oslash \mathbf{n}_\times$



Attention in GNN Models – Forward Pass

Formulating ψ is the „crux" of devising a concrete formulation for a specific model

$\mathbf{H}^{l+1} =$ 

$$\mathbf{H}^{l+1} = \sigma\left(\mathbf{Z}\right), \quad \mathbf{Z} = \Psi\mathbf{H}\mathbf{W}$$

weights

Non-linearity

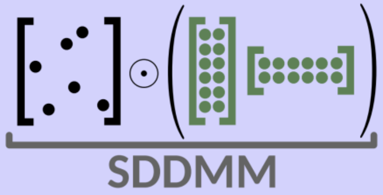
A sparse *n x n* tensor with attention scores, **model specific**

Features from previous layer

Vanilla Attention

Ψ

$$\Psi = \mathcal{A} \odot \mathbf{H}_{\times}$$



Graph Attention Network (GAT)

Ψ ★

$$\Psi = \mathrm{sm}(\mathcal{T}) \quad \mathcal{T} = \mathcal{A} \odot \exp\left(\sigma\left(\mathbf{C}\right)\right)$$

$$\mathbf{C} = \mathrm{rep}_n^T\left((\mathbf{H}'\mathbf{a})^T\right) + \mathrm{rep}_n(\mathbf{H}'\overline{\mathbf{a}})$$

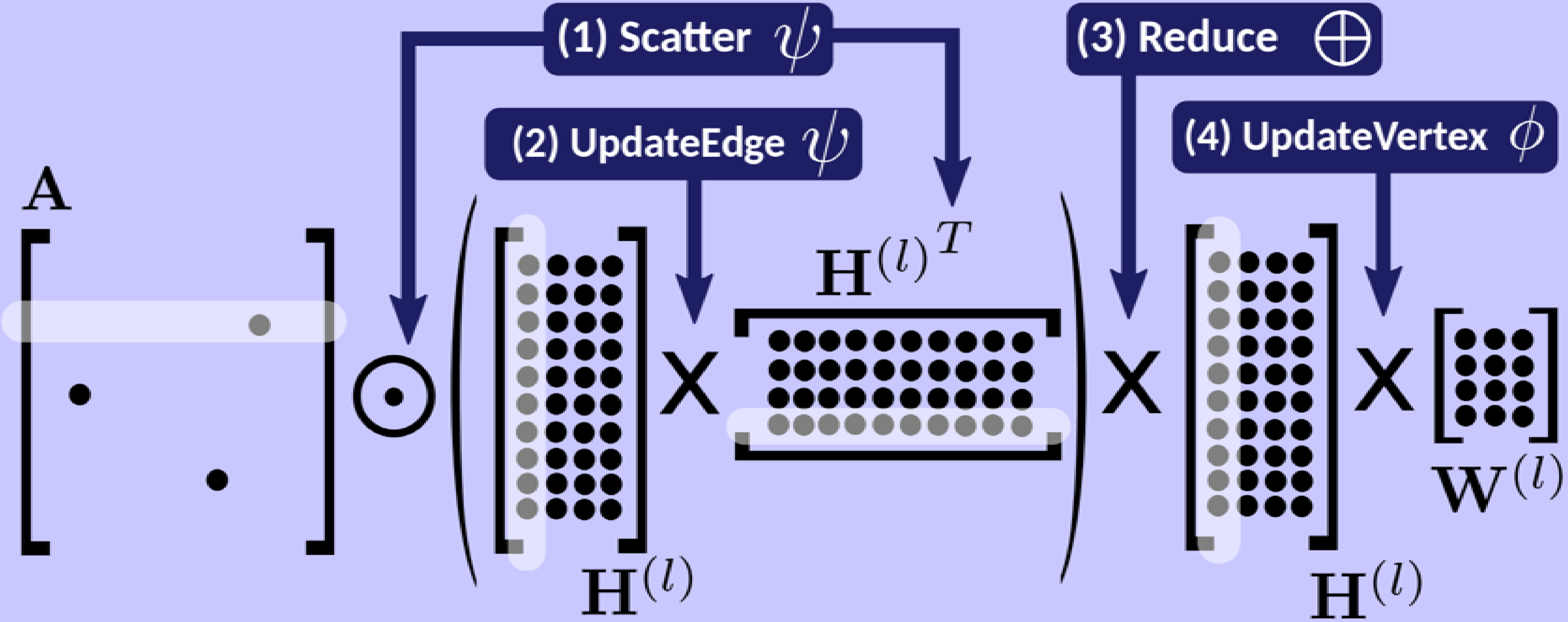
Attention-based GNN (AGNN)

Ψ ★

$$\Psi = \mathcal{A} \odot \mathbf{H}_{\times} \oslash \mathbf{n}_{\times}$$

Global Formulations of GNN Models

Example model: Graph Attention Network based on Dot Product (Vanilla Attention)

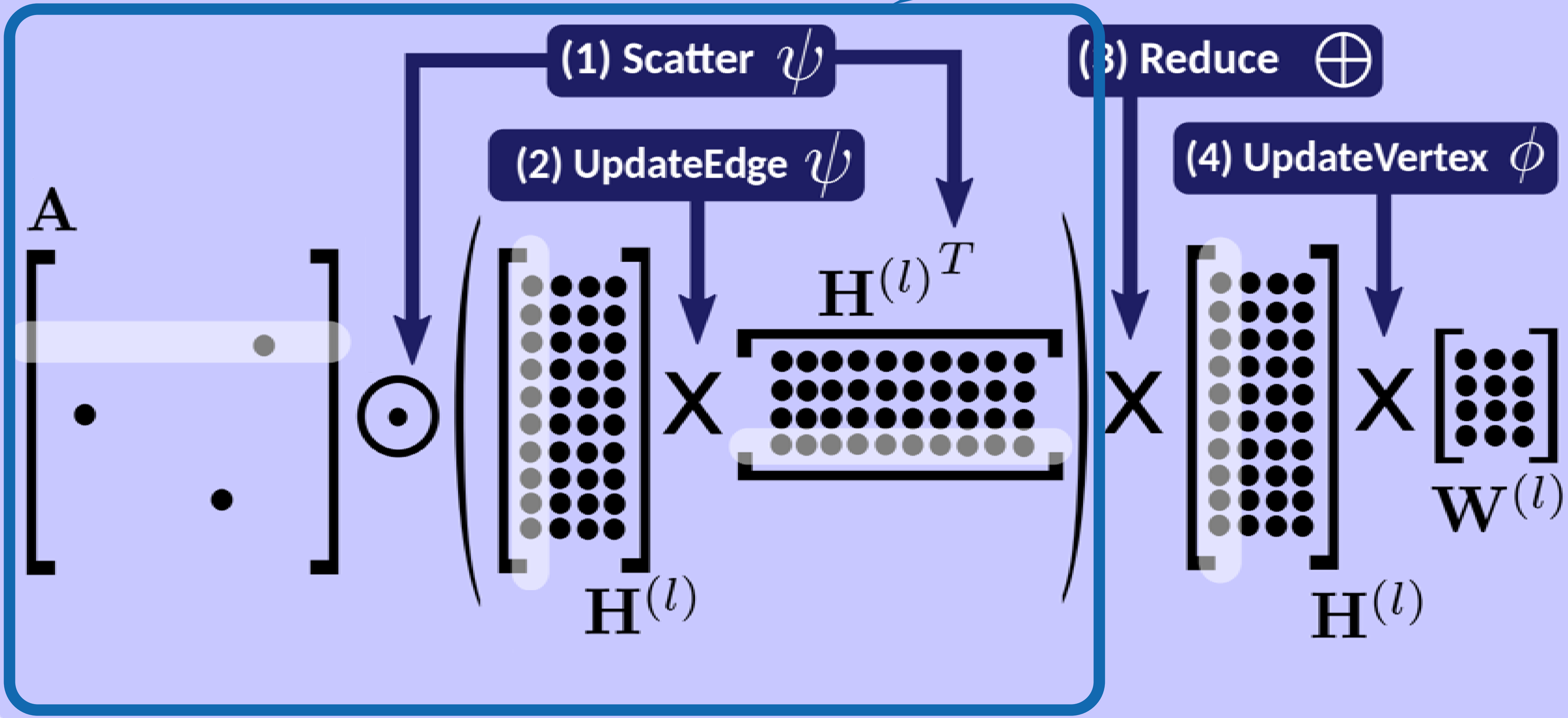


A. Vaswani, et al. Attention is All you Need. NIPS. 2017

Global Formulations of GNN Models

This is our ψ for the Vanilla Attention Model

Example model: Graph Attention Network based on Dot Product (Vanilla Attention)



A. Vaswani, et al. Attention is All you Need. NIPS. 2017

Attention in GNN Models – Forward Pass

Formulating ψ is the „crux" of devising a concrete formulation for a specific model

$$\mathbf{H}^{l+1} = \sigma\left(\mathbf{Z}\right), \quad \mathbf{Z} = \Psi\mathbf{H}\mathbf{W}$$

weights

Non-linearity

A sparse *n x n* tensor with attention scores, **model specific**

Features from previous layer

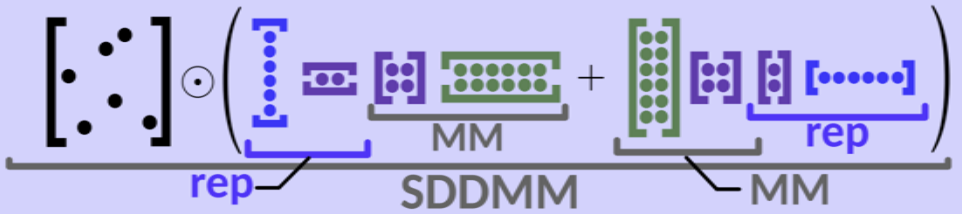
Vanilla Attention

Ψ

$$\Psi = \mathcal{A} \odot \mathbf{H}_{\times}$$

Graph Attention Network (GAT)

Ψ ★



$$\Psi = \mathrm{sm}(\mathcal{T}) \quad \mathcal{T} = \mathcal{A} \odot \exp\left(\sigma\left(\mathbf{C}\right)\right)$$

$$\mathbf{C} = \mathrm{rep}_n^T((\mathbf{H}'\underline{\mathbf{a}})^T) + \mathrm{rep}_n(\mathbf{H}'\overline{\mathbf{a}})$$

Attention-based GNN (AGNN)

Ψ ★

$$\Psi = \mathcal{A} \odot \mathbf{H}_{\times} \oslash \mathbf{n}_{\times}$$

Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

Vector concatenation

P. Veličković, et al. Graph Attention Networks. ICLR . 2018.

Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

This is the softmax normalization, we'll get to it later ☺

P. Veličković, et al. Graph Attention Networks. ICLR . 2018.

Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

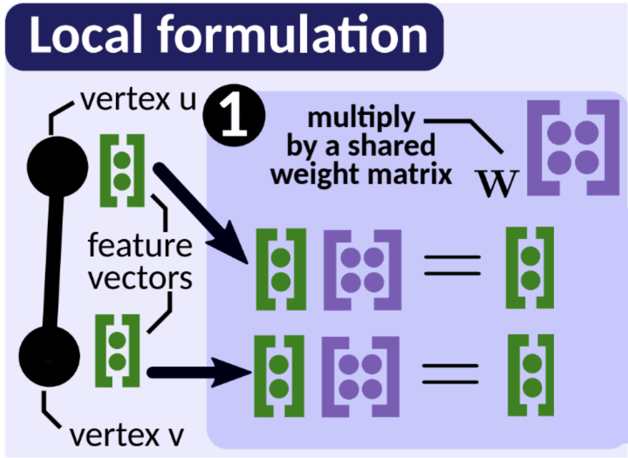
$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

P. Veličković, et al. Graph Attention Networks. ICLR . 2018.

Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

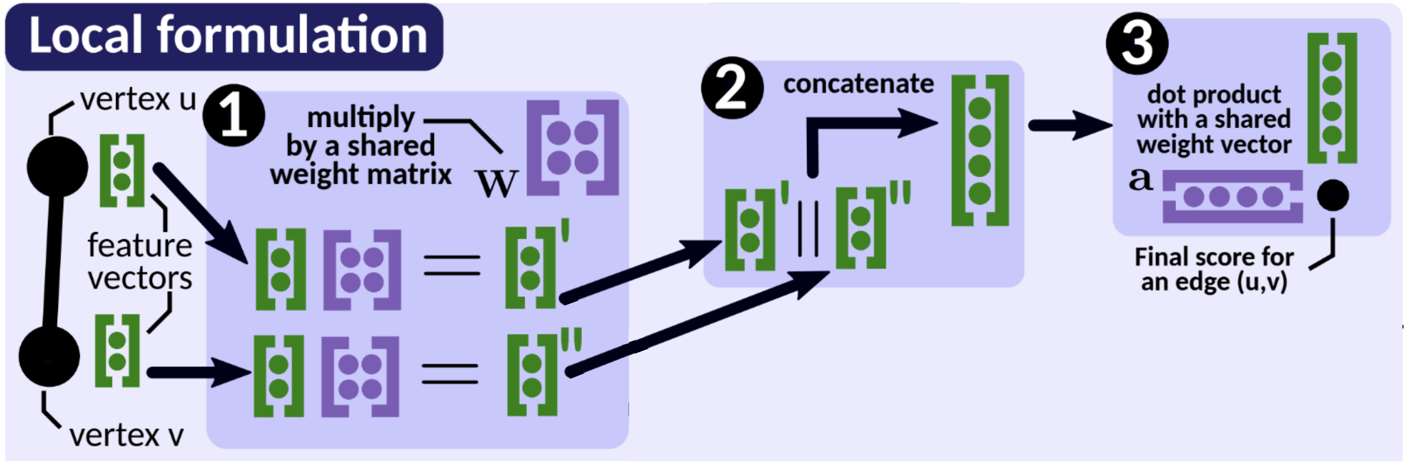
$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



Graph Attention Network (GAT)

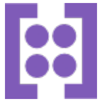
Local ψ formulation is very involving – how to obtain the global formulation?

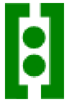
$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

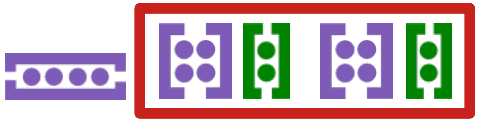


Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

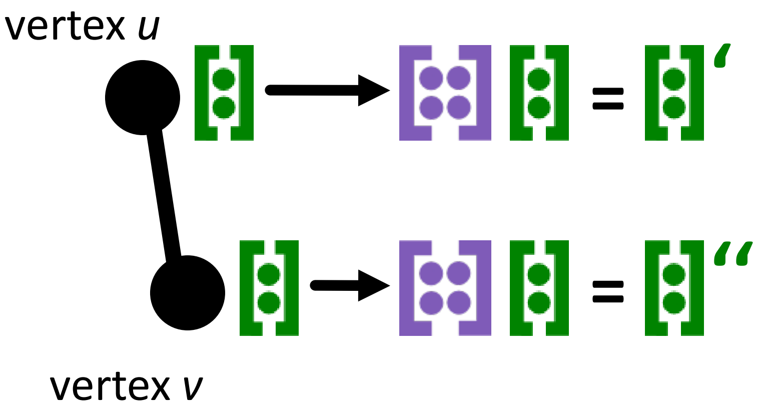
 weight matrix  weight vector

 feature vector



$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

Local formulation



multiply by shared weight matrix

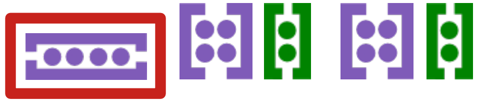
Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

weight matrix

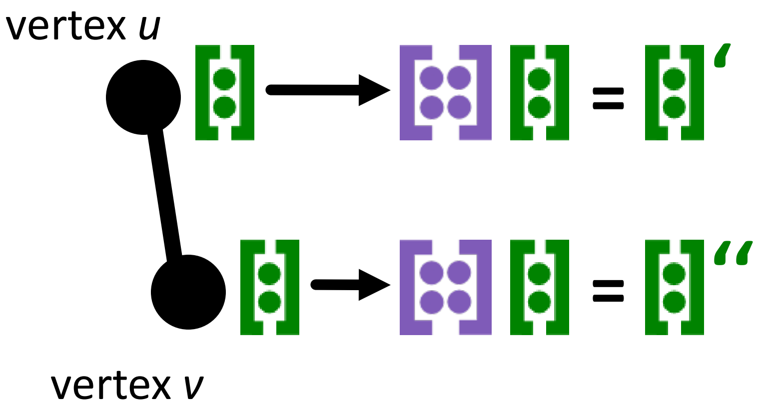
weight vector

feature vector

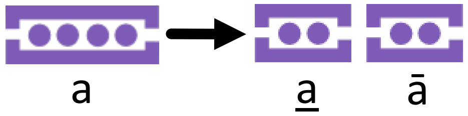


$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

Local formulation



multiply by shared weight matrix



vector split

Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

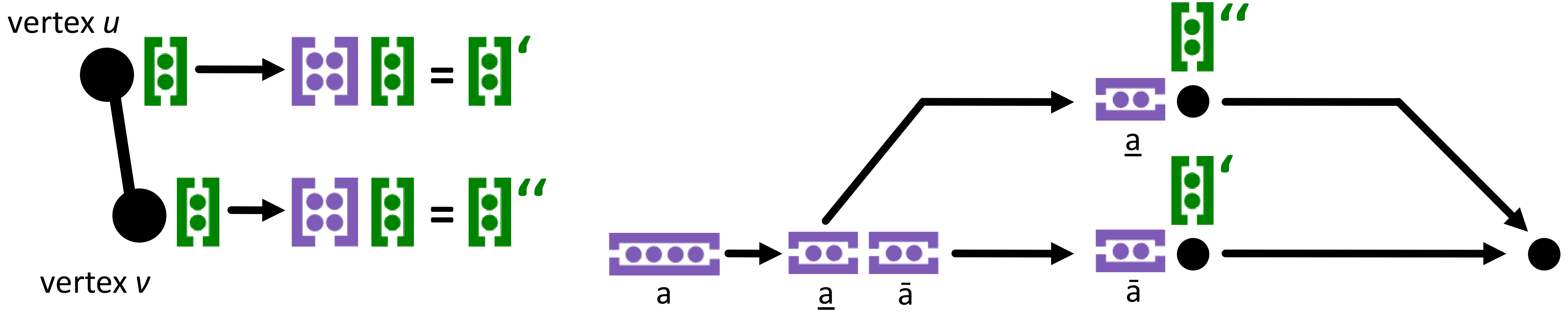
weight matrix

weight vector

feature vector

Local formulation

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



multiply by shared weight matrix

vector split

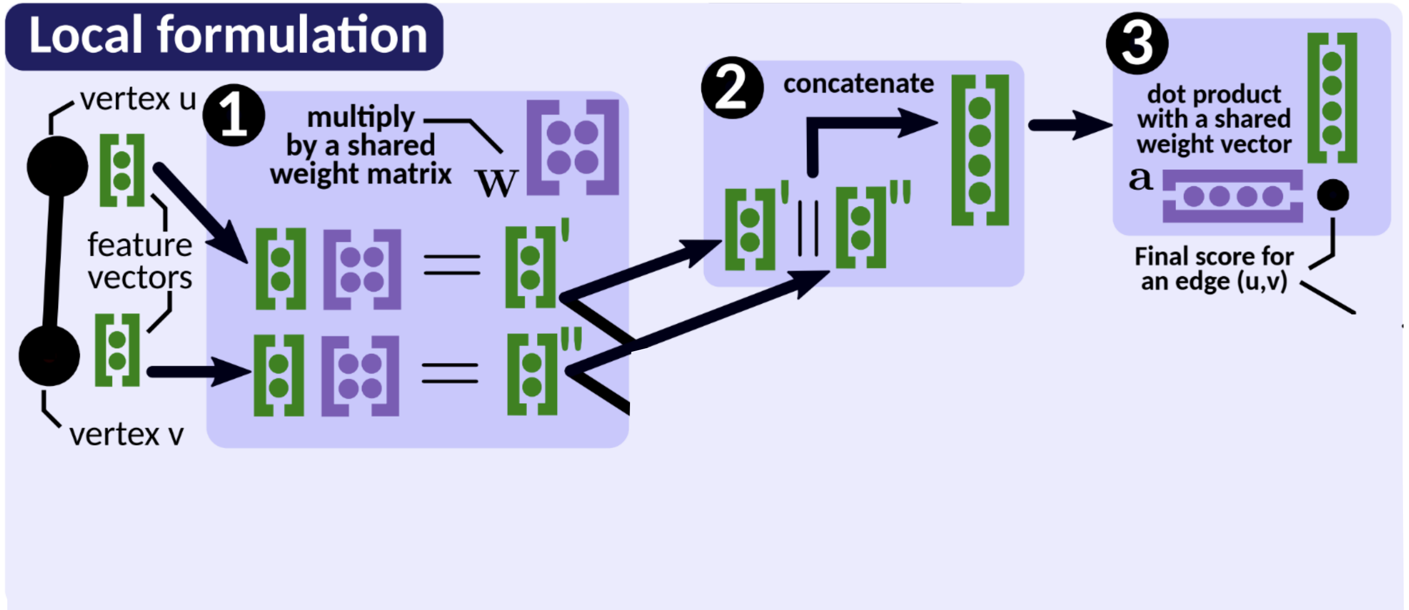
two dot products

sum partial sums

Graph Attention Network (GAT)

Local ψ formulation is very involving – how to obtain the global formulation?

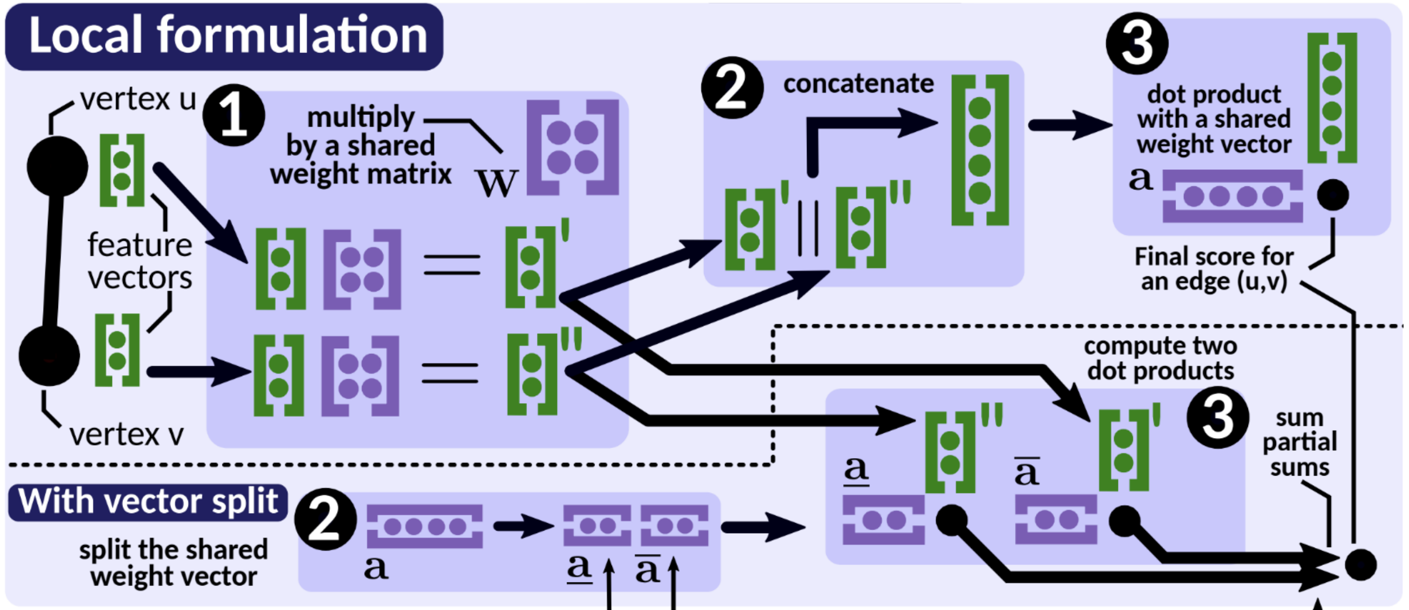
$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



Graph Attention Network (GAT)

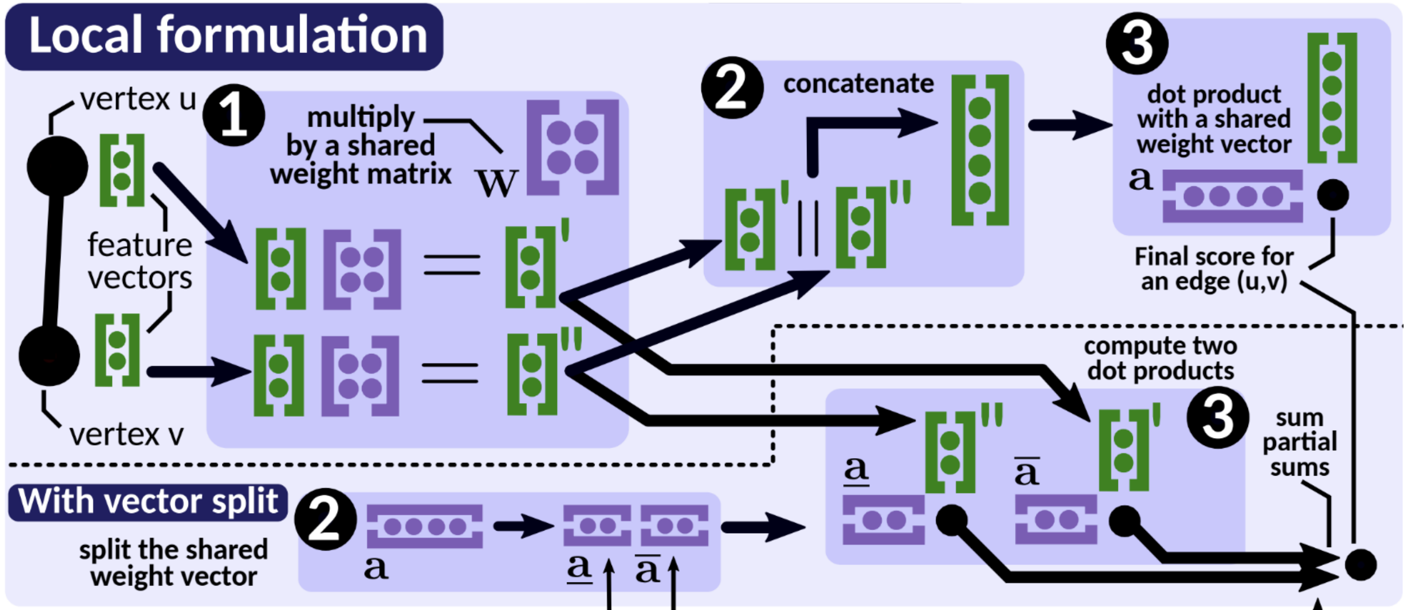
Local ψ formulation is very involving – how to obtain the global formulation?

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



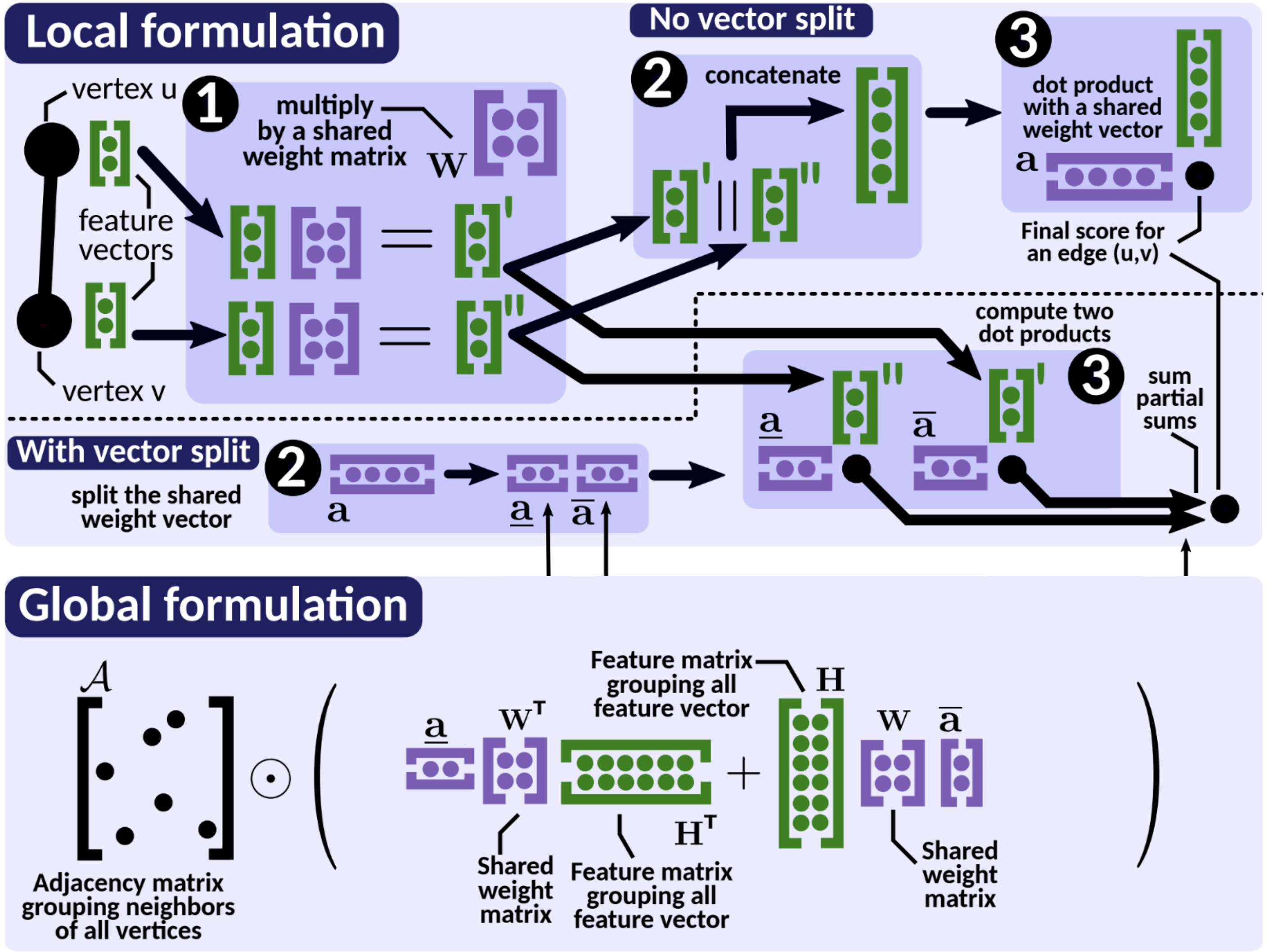
Graph Attention Network (GAT)

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



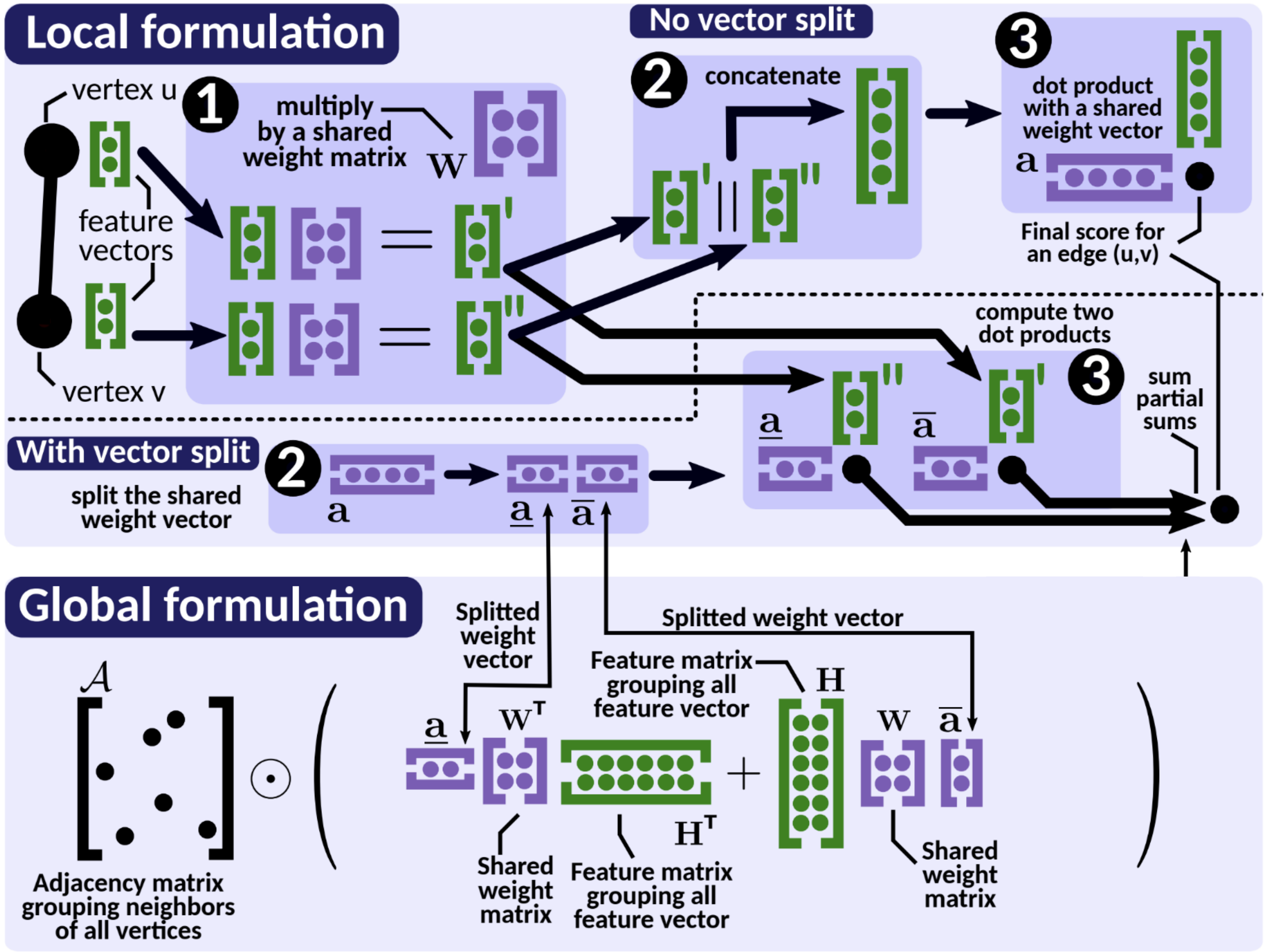
Graph Attention Network (GAT)

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \middle\| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \middle\| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



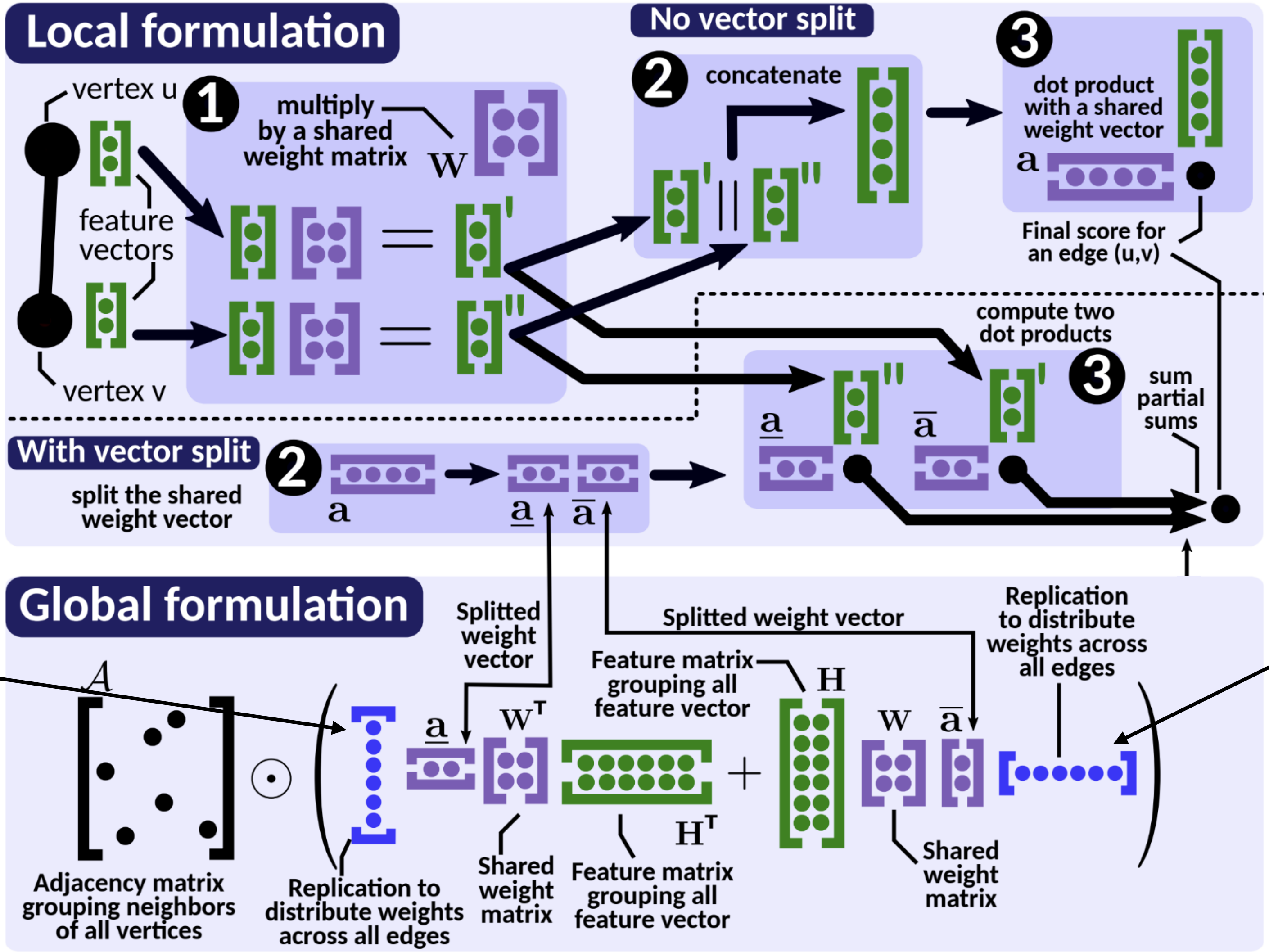
Graph Attention Network (GAT)

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



Graph Attention Network (GAT)

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$



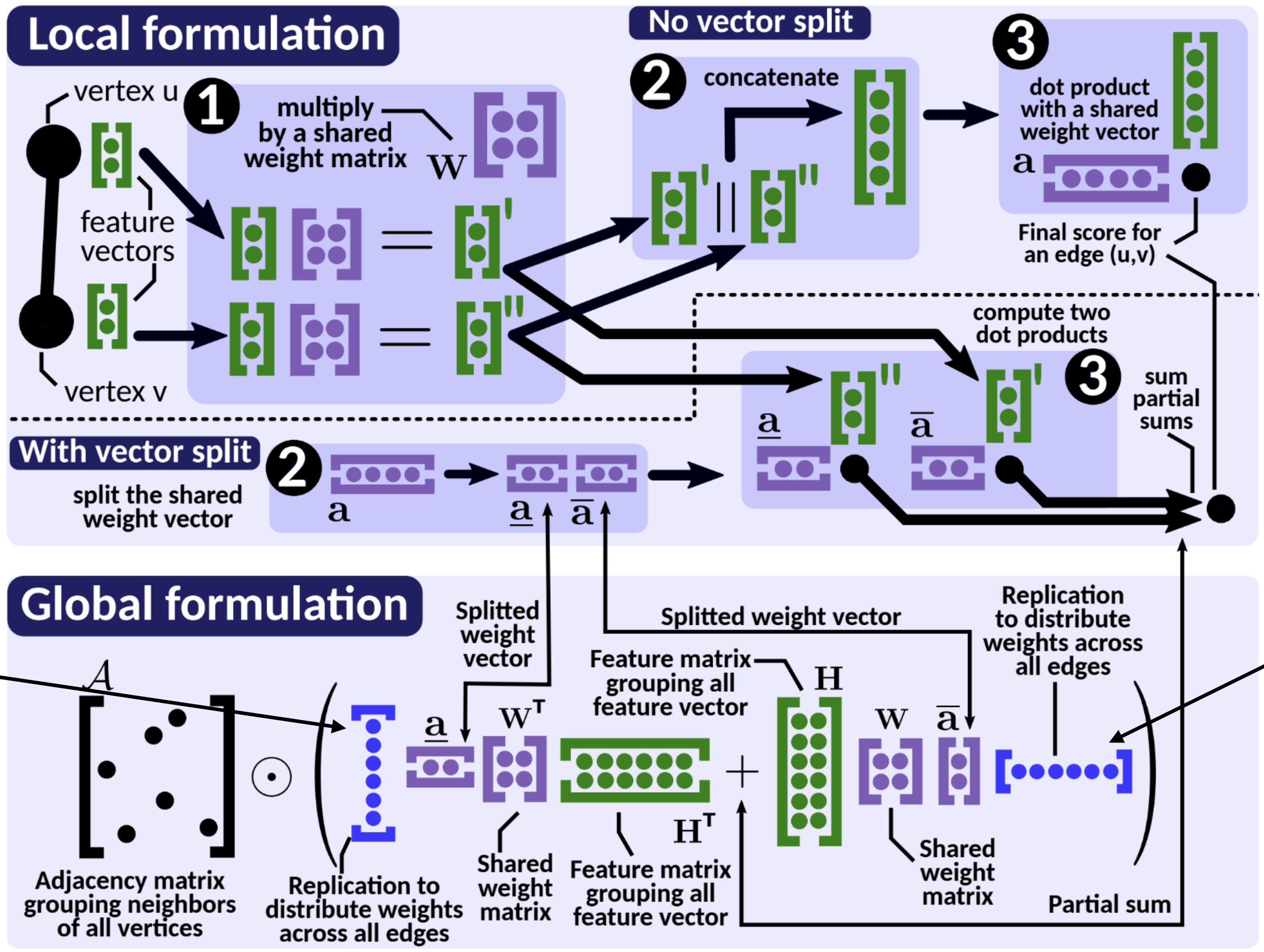
$$\mathrm{rep}_n^T((\mathbf{H}'\underline{\mathbf{a}})^T)$$

$$\mathrm{rep}_n(\mathbf{H}'\overline{\mathbf{a}})$$

Graph Attention Network (GAT)

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \middle\| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \widehat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \middle\| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

$\mathrm{rep}_n^T((\mathbf{H}'\underline{\mathbf{a}})^T)$

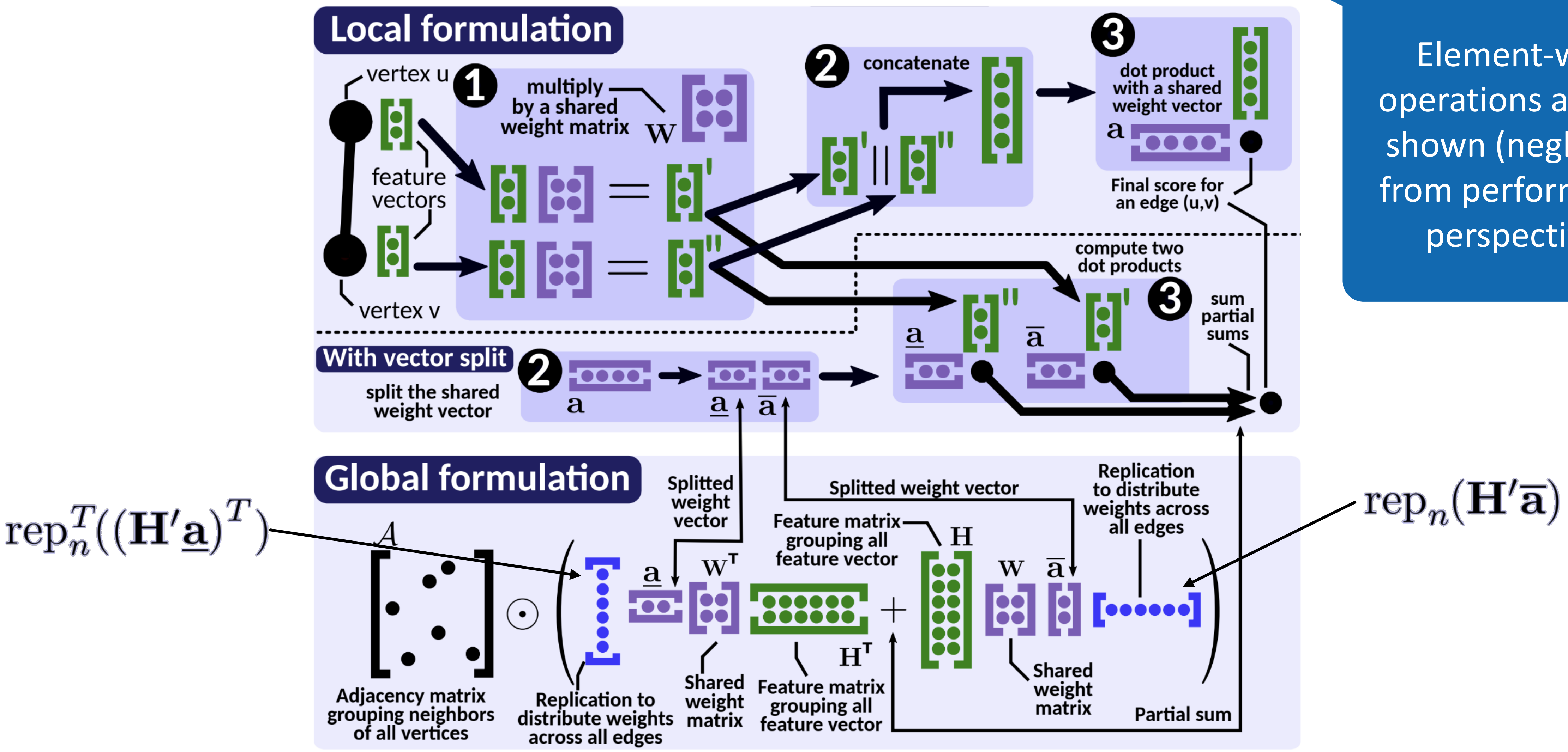


$\mathrm{rep}_n(\mathbf{H}'\overline{\mathbf{a}})$

Graph Attention Network (GAT)

$$\psi_{v,u} = \frac{\exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_u\right]\right)\right)}{\sum_{y \in \hat{N}(v)} \exp\left(\sigma\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_y\right]\right)\right)} \mathbf{h}_u$$

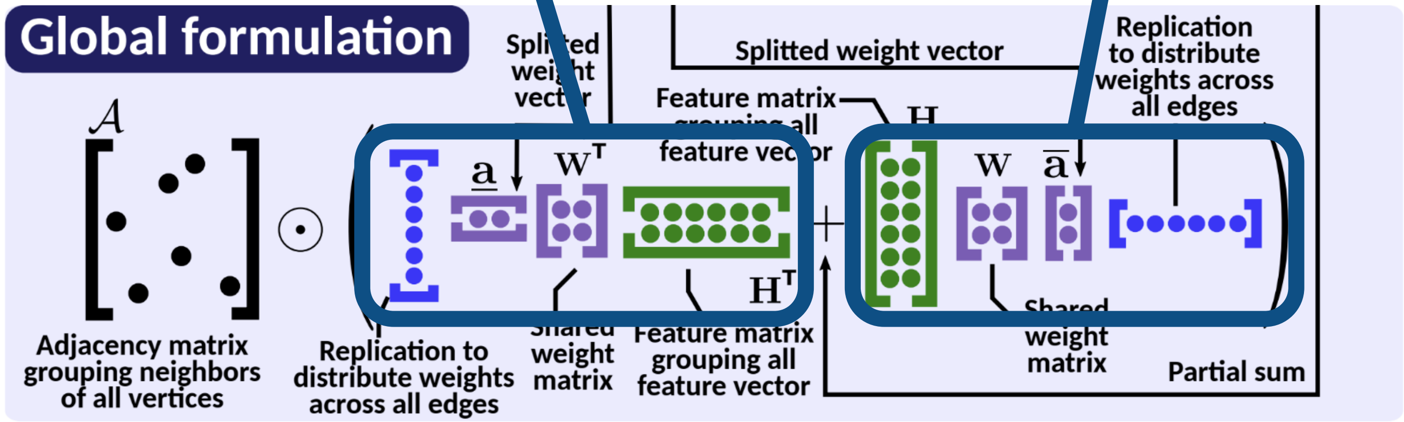
Element-wise operations are not shown (negligible from performance perspective)



Graph Attention Network (GAT)

$$\Psi = \mathrm{sm}(\mathcal{X}) \quad \mathcal{X} = \mathcal{A} \odot \exp\left(\sigma\left(\mathbf{C}\right)\right)$$

$$\mathbf{C} = \mathrm{rep}_n^T((\mathbf{H}'\underline{\mathbf{a}})^T) + \mathrm{rep}_n(\mathbf{H}'\overline{\mathbf{a}})$$

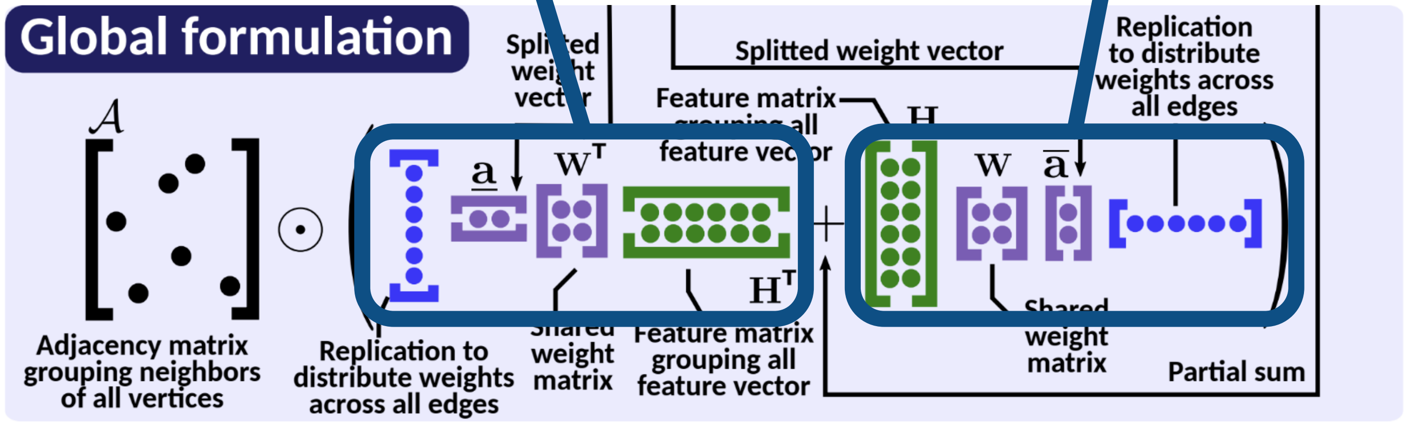


Graph Attention Network (GAT)

Let's see how **softmax** (sm) looks like in the global formulation

$$\Psi = \boxed{\mathrm{sm}(\mathcal{X})} \quad \mathcal{X} = \mathcal{A} \odot \exp\left(\sigma\left(\mathbf{C}\right)\right)$$

$$\mathbf{C} = \mathrm{rep}_n^T((\mathbf{H}'\underline{\mathbf{a}})^T) + \mathrm{rep}_n(\mathbf{H}'\overline{\mathbf{a}})$$



Global Formulations of GNN Kernels – Softmax

Tensor algebra expression

$$\mathrm{sm}(\mathcal{X}) = \exp(\mathcal{X}) \oslash \boxed{\mathrm{rs}_n}(\exp(\mathcal{X})) \\ = \exp(\mathcal{X}) \oslash (\exp(\mathcal{X})\ \mathbf{1}\mathbf{1}^T)$$

part 1 of rs()

Example

$$\mathcal{X} = \begin{bmatrix} 0&0&0&2&1 \\ 0&0&1&0&0 \\ 0&1&0&0&4 \\ 2&0&0&0&3 \\ 1&0&4&3&0 \end{bmatrix} \xrightarrow[\text{(1) element-wise exponentiation of non-zero elements}]{\mathcal{X}_1 = \exp(\mathcal{X})} \begin{bmatrix} 0&0&0&e^2&e \\ 0&0&e&0&0 \\ 0&e&0&0&e^4 \\ e^2&0&0&0&e^3 \\ e&0&e^4&e^3&0 \end{bmatrix} \xrightarrow[\text{(2) multiplication by a column vector of ones}]{\mathcal{X}_2 = \mathrm{sum}(\mathcal{X}_1)} \begin{bmatrix} 0&0&0&e^2&e \\ 0&0&e&0&0 \\ 0&e&0&0&e^4 \\ e^2&0&0&0&e^3 \\ e&0&e^4&e^3&0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} e^2+e \\ e \\ e+e^4 \\ e^2+e^3 \\ e+e^4+e^3 \end{bmatrix}$$

part 2 of rs()

$$\xrightarrow[\text{(3) multiplication by a row vector of ones}]{\mathcal{X}_3 = \mathrm{rep}_n(\mathcal{X}_2)} \begin{bmatrix} e^2+e \\ e \\ e+e^4 \\ e^2+e^3 \\ e+e^4+e^3 \end{bmatrix} \begin{bmatrix} 1&1&1&1&1 \end{bmatrix} = \begin{bmatrix} e^2+e & \cdots \\ e & \\ e+e^4 & \\ e^2+e^3 & \\ e+e^4+e^3 & \end{bmatrix}$$

Global Formulations of GNN Kernels – Softmax

Tensor algebra expression

$$\mathrm{sm}(\mathcal{X}) = \exp(\mathcal{X}) \oslash \mathrm{rs}_n(\exp(\mathcal{X}))$$
$$= \exp(\mathcal{X}) \oslash \left(\exp(\mathcal{X})\ \mathbf{1}\mathbf{1}^T\right)$$

Example

$$\mathcal{X} = \begin{bmatrix} 0&0&0&2&1\\ 0&0&1&0&0\\ 0&1&0&0&4\\ 2&0&0&0&3\\ 1&0&4&3&0 \end{bmatrix} \xrightarrow[\text{(1) element-wise exponentiation of non-zero elements}]{\mathcal{X}_1 = \exp(\mathcal{X})} \begin{bmatrix} 0&0&0&e^2&e\\ 0&0&e&0&0\\ 0&e&0&0&e^4\\ e^2&0&0&0&e^3\\ e&0&e^4&e^3&0 \end{bmatrix} \xrightarrow[\text{(2) multiplication by a column vector of ones}]{\mathcal{X}_2 = \mathrm{sum}(\mathcal{X}_1)} \begin{bmatrix} 0&0&0&e^2&e\\ 0&0&e&0&0\\ 0&e&0&0&e^4\\ e^2&0&0&0&e^3\\ e&0&e^4&e^3&0 \end{bmatrix} \begin{bmatrix} 1\\1\\1\\1\\1 \end{bmatrix} = \begin{bmatrix} e^2+e\\ e\\ e+e^4\\ e^2+e^3\\ e+e^4+e^3 \end{bmatrix}$$

$$\xrightarrow[\text{(3) multiplication by a row vector of ones}]{\mathcal{X}_3 = \mathrm{rep}_n(\mathcal{X}_2)} \begin{bmatrix} e^2+e\\ e\\ e+e^4\\ e^2+e^3\\ e+e^4+e^3 \end{bmatrix} \begin{bmatrix} 1&1&1&1&1 \end{bmatrix} = \begin{bmatrix} e^2+e & \\ e & \\ e+e^4 & \cdots \\ e^2+e^3 & \\ e+e^4+e^3 & \end{bmatrix} \xrightarrow[\text{(4) element-wise division of edge scores (normalization)}]{\mathcal{X}_1 \oslash \mathcal{X}_3} \begin{bmatrix} 0&0&0&e^2&e\\ 0&0&e&0&0\\ 0&e&0&0&e^4\\ e^2&0&0&0&e^3\\ e&0&e^4&e^3&0 \end{bmatrix} \oslash \begin{bmatrix} e^2+e & \\ e & \\ e+e^4 & \cdots \\ e^2+e^3 & \\ e+e^4+e^3 & \end{bmatrix}$$

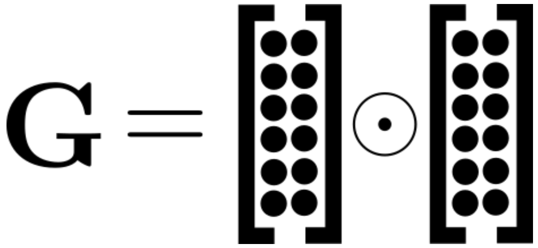
Global Formulations of GNN Kernels – Backward Pass

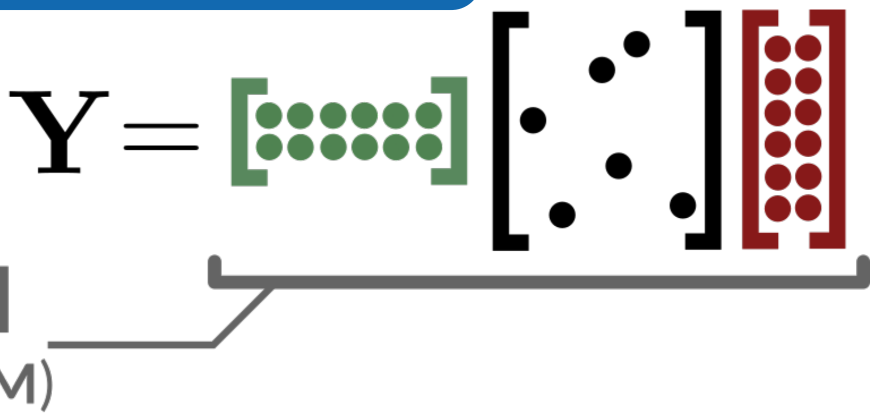
Generic formulation

$$\mathbf{G}^{l-1} = \sigma'\left(\mathbf{Z}^{l-1}\right) \odot \Gamma^{l}$$

$$\mathbf{Y}^{l} = {\mathbf{H}^{l}}^{T}\,\Psi\left(\mathcal{A}^{T}, \mathbf{H}^{l}\right)\mathbf{G}^{l} + \mathbf{G}^{l}{\mathbf{W}^{l}}^{T}{\mathbf{H}^{l}}^{T}\,\frac{\partial\Psi}{\partial\mathbf{W}^{l}}$$

Matrix view





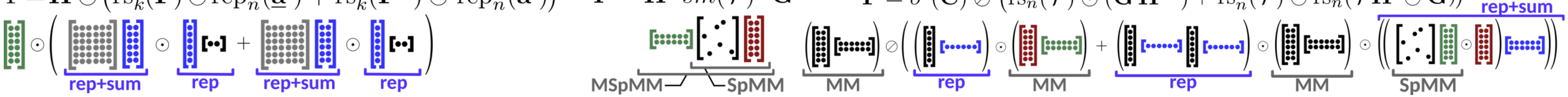
Global Formulations of GNN Kernels – Backward Pass

Vanilla Attention (VA)

$\Gamma = \mathbf{N}_+\mathbf{H} + (\mathcal{A}^T \odot \mathbf{H}_\times)\,\mathbf{M}$ $\quad \mathbf{N} = \mathcal{A} \odot (\mathbf{M}\mathbf{H}^T)$ $\quad \mathbf{M} = \mathbf{G}\,\mathbf{W}^T$ $\quad \mathbf{Y} = \mathbf{H}^T\,(\mathcal{A}^T \odot \mathbf{H}_\times)\,\mathbf{G}$

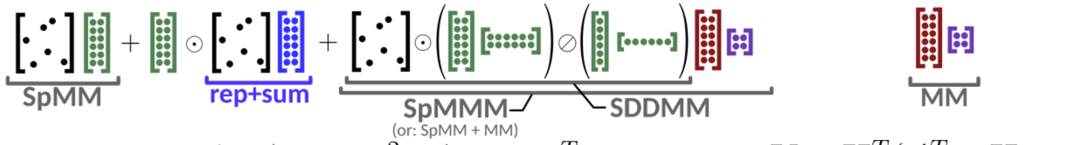
Graph Attention Network (GAT)

$\Gamma = \mathbf{H} \odot (\mathrm{rs}_k(\mathbf{F}) \odot \mathrm{rep}_n^T(\underline{\mathbf{a}}') + \mathrm{rs}_k(\mathbf{F}^T) \odot \mathrm{rep}_n^T(\overline{\mathbf{a}}'))$ $\quad \mathbf{Y} = \mathbf{H}^T sm(\mathcal{T})^T\mathbf{G}$ $\quad \mathbf{F} = \sigma'(\mathbf{C}) \oslash (\mathrm{rs}_n(\mathcal{T}) \odot (\mathbf{G}\,\mathbf{H}'^T) + \mathrm{rs}_n^2(\mathcal{T}) \odot \mathrm{rs}_n(\mathcal{T}\mathbf{H}' \odot \mathbf{G}))$

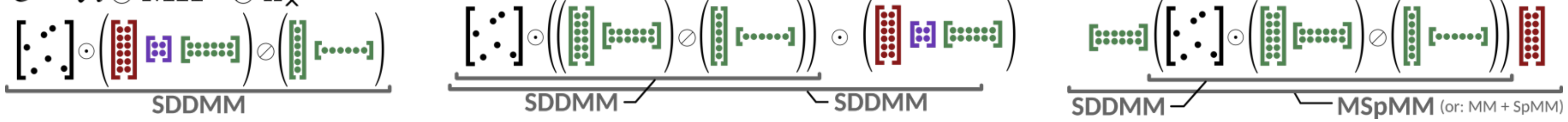


Attention-based GNN (AGNN) [1]

$\Gamma = \mathcal{S}_+\mathbf{H} + \mathbf{H} \odot \mathrm{rs}_k(\mathcal{R}_+) + (\mathcal{A}^T \odot \mathbf{H}_\times \oslash \mathbf{n}_\times)\mathbf{M}$ $\quad \mathbf{M} = \mathbf{G}\,\mathbf{W}^T$



$\mathcal{S} = \mathcal{A} \odot \mathbf{M}\mathbf{H}^T \oslash \mathbf{n}_\times$ $\quad \mathcal{R} = \mathcal{A} \odot (\mathbf{H}_\times \oslash \mathbf{n}^2\mathbf{n}_\times) \odot \mathbf{M}\mathbf{H}^T$ $\quad \mathbf{Y} = \mathbf{H}^T(\mathcal{A}^T \odot \mathbf{H}_\times \oslash \mathbf{n}_\times)\mathbf{G}$



[1] K. Thekumparampil et al. Attention-based Graph Neural Network for Semi-supervised Learning. arXiv:2018.

The Entire Optimization Toolchain

A GNN model local formulation

A formulation provided by a user, e.g., a GNN model designer

$$\mathbf{h}_i^{l+1} = \phi\left(\mathbf{h}_i^l, \bigoplus_{j\in N(i)} \psi\left(\mathbf{h}_i^l, \mathbf{h}_j^l\right)\right)$$

A GNN model global formulation

A global formulation designed using techniques from this work (Sections 3-5)

$$\mathbf{H}^{l+1} = \sigma\left(\left(\Phi \circ \oplus\right)\left(\Psi\left(\mathcal{A}, \mathbf{H}^l\right), \mathbf{H}^l\right)\right)$$

$$\mathbf{Y}^l = \mathbf{H}^{l^T} \Psi\left(\mathcal{A}^T, \mathbf{H}^l\right) \mathbf{G}^l$$

$$\mathbf{G}^l = \sigma'\left(\mathbf{Z}^{l-1}\right) \odot \Gamma^{l-1}$$

Execution: manual

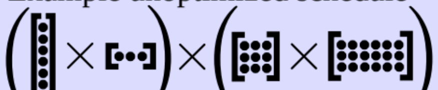
Formulation optimization

An optimization of the formulation, aiming to reduce communication

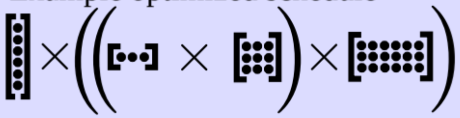
Execution: SOAP framework

Example formulation

Example unoptimized schedule



Example optimized schedule



High-level implementation

An implementation of the model, potentially specifying details of parallelism or distribution

Execution: Manual, or using a library or a DSL (e.g., GraphBLAS, GraphBLAST, Combinatorial BLAS, GraphMat, Cyclops Tensor Framework)

Implementation optimizations

Fusing different kernels, applying communication minimizing schemes

Execution: Manual, based on the suggestions proposed in this work

Code tuning

Code tuning, e.g., enhancing vectorizability

Execution: Manual

Blue color: [this work]

Grey color: existing work

Minimizing communication = Maximizing data reuse = Data reuse within a kernel: ***tiling*** + Data reuse across kernels: ***kernel fusion***

SOAP: G. Kwasniewski et al. Pebbles, Graphs, and a Pinch of Combinatorics: Towards Tight I/O Lower Bounds for Statically Analyzable Programs. SPAA. 2021.

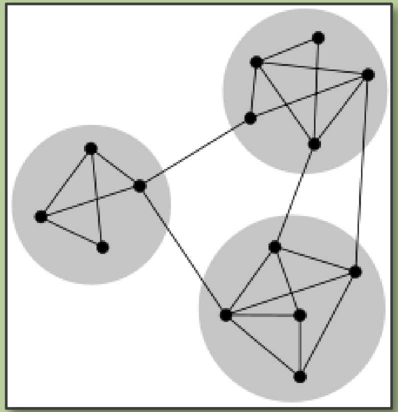
Evaluation

CSCS Cray Piz Daint supercomputer

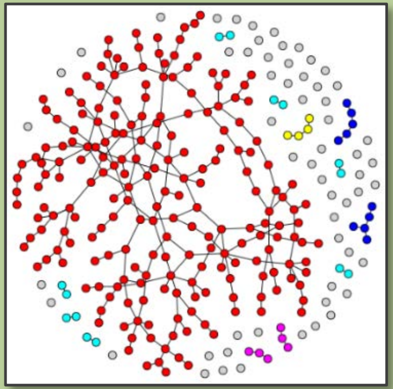
- Cray XC50 nodes
- Intel Xeon E5-2690 v3, 12 cores
- Single NVIDIA Tesla P100 per node
- 64 GB RAM per node

Considered Graph Datasets

Synthetic graphs

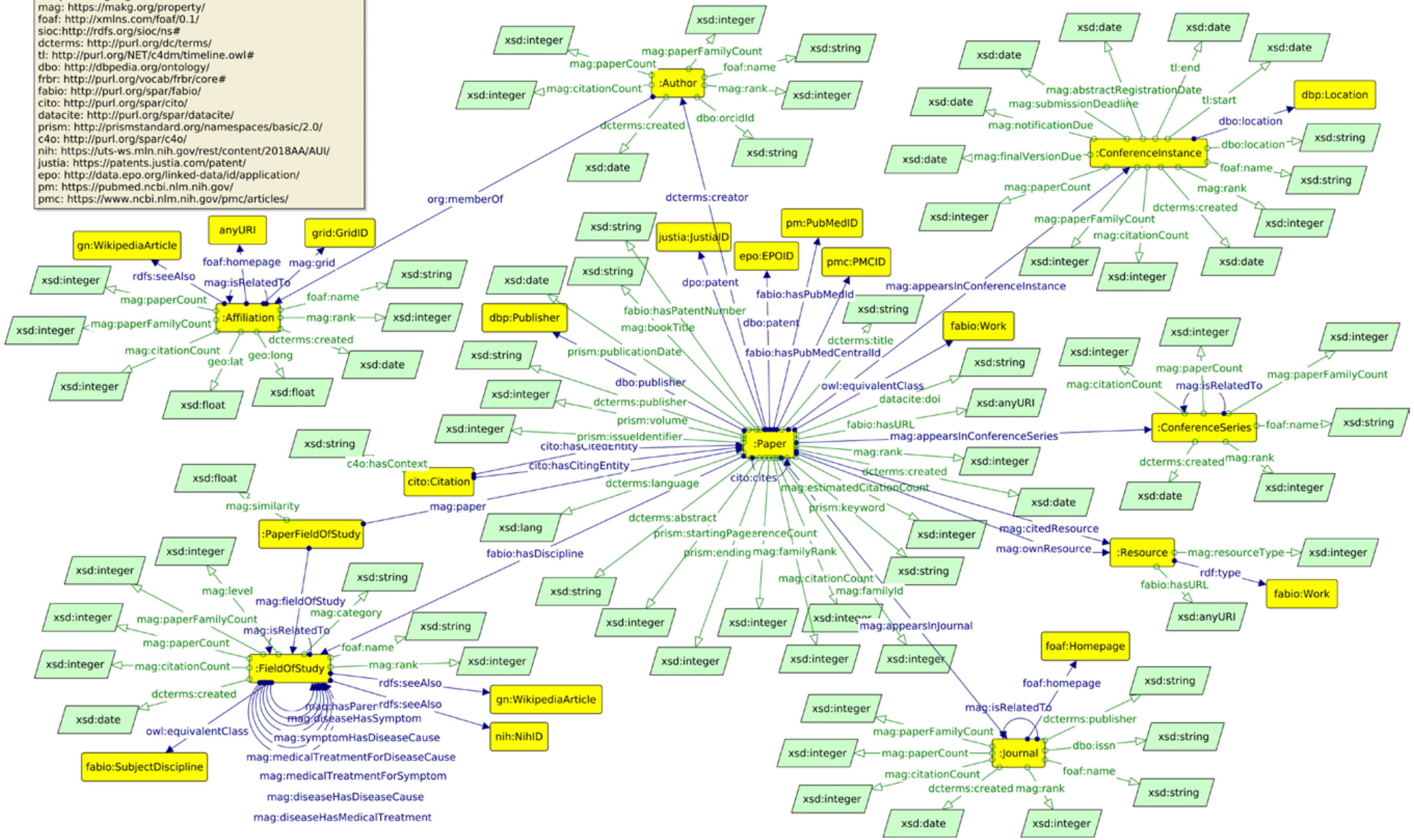


Kronecker [1]



Erdös-Rényi [2]

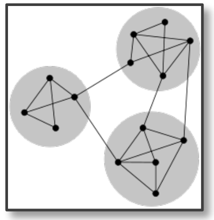
Microsoft Academic Knowledge Graph



[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.
[2] P. Erdos and A. Renyi. On the evolution of random graphs. Pub. Math. Inst. Hun. A. Science. 1960.

Strong Scaling

Kronecker [1]



—o—GAT - This Work - -o- -GAT - DistDGL ● GAT - DGL

—◆—VA - This Work - -◆- -VA - DistDGL ◆ VA - DGL

—□—AGNN - This Work - -□- -AGNN - DistDGL ■ AGNN - DGL

p = 1%

p = 0.01%

Sparsity: p = m / (n * n)

k=16

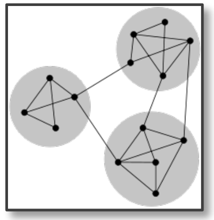
#features

k=128

[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.

Strong Scaling

Kronecker [1]



—o— GAT - This Work
—◇— VA - This Work
—□— AGNN - This Work
- -o- - GAT - DistDGL
- -◇- - VA - DistDGL
- -□- - AGNN - DistDGL
● GAT - DGL
♦ VA - DGL
■ AGNN - DGL

p = 1%

n = 131k, m = 172M

n = 262k, m = 687M

p = 0.01%

n = 1M, m = 110M

n = 2.1M, m = 440M

#vertices

#edges

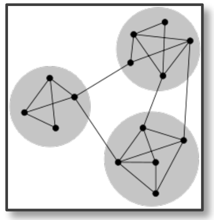
k=16

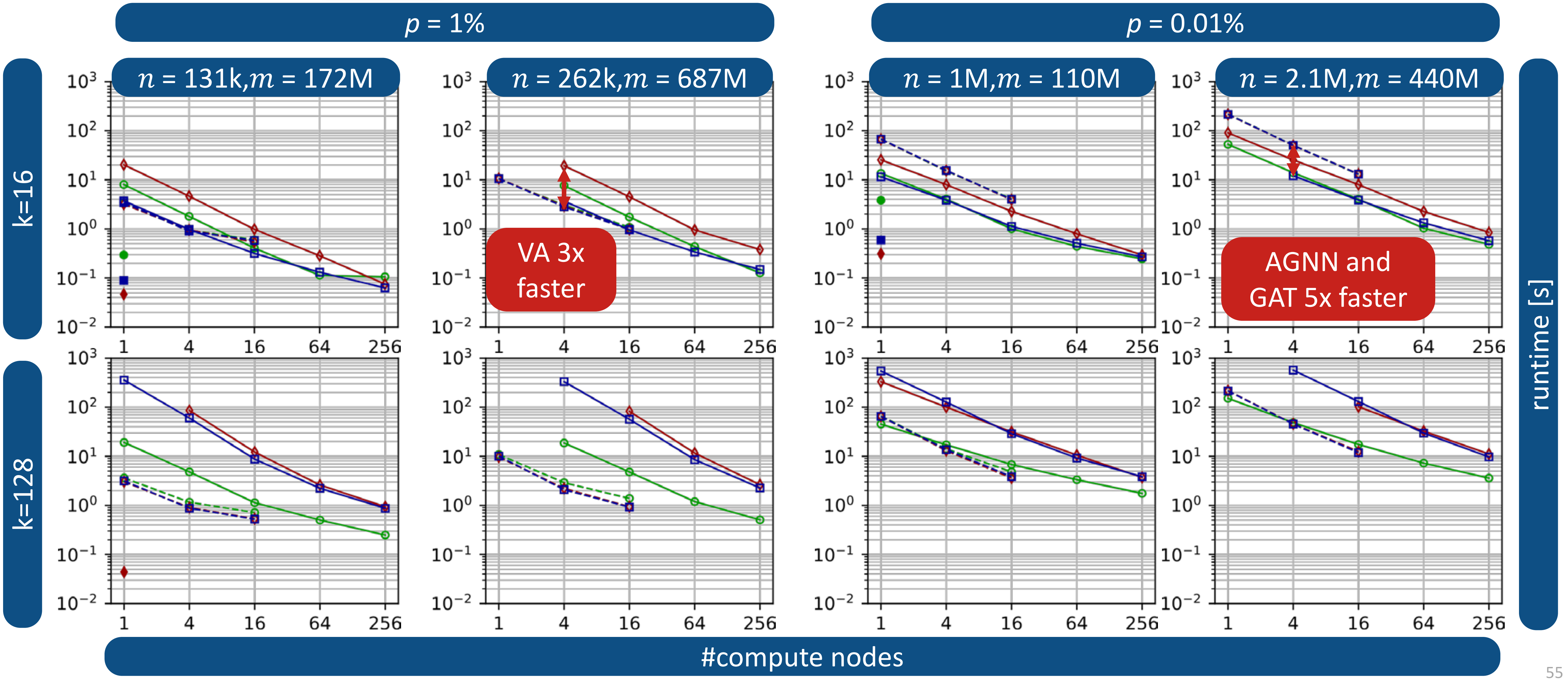
k=128

[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.

Strong Scaling

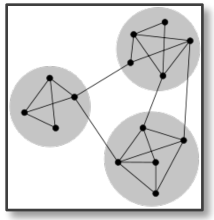
Kronecker [1]

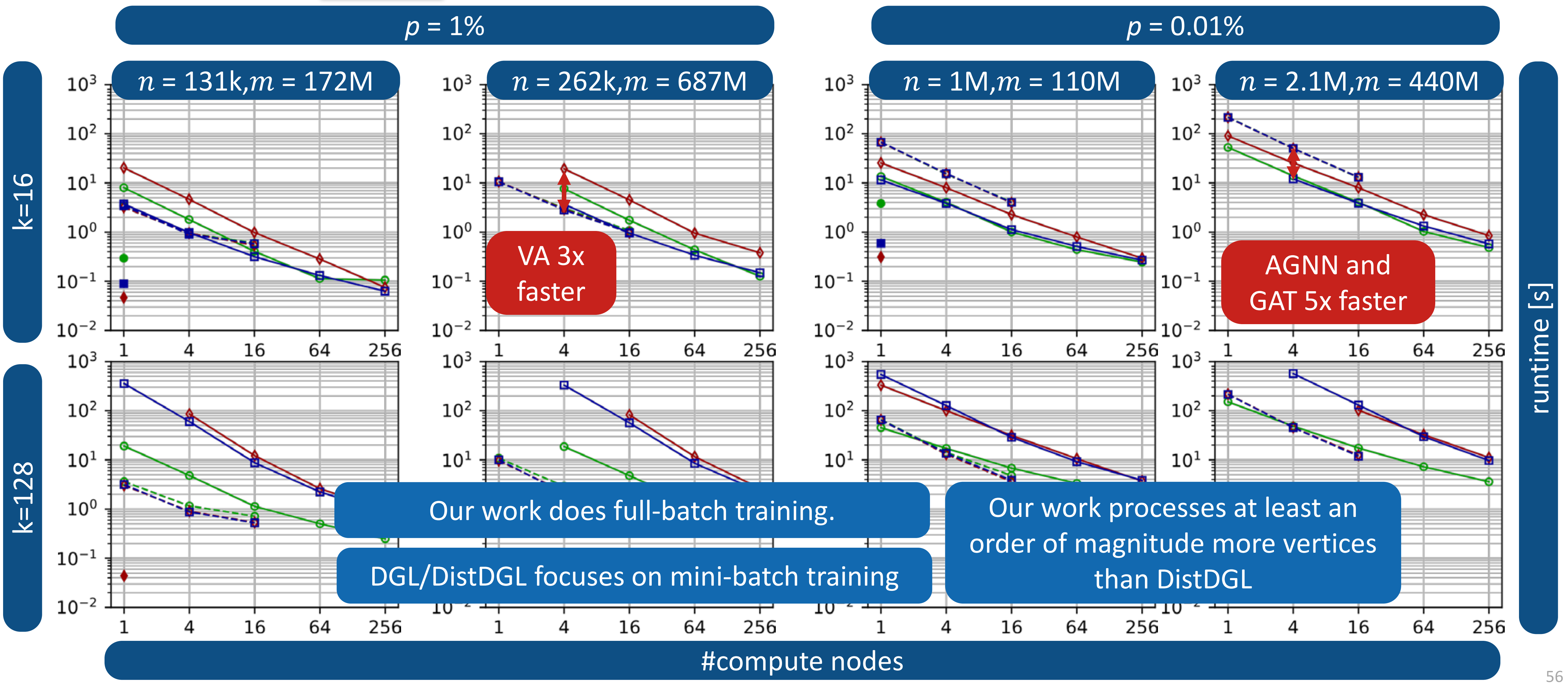




Strong Scaling

Kronecker [1]





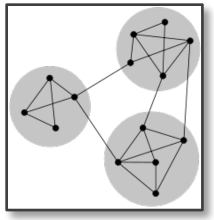
Our work does full-batch training.

DGL/DistDGL focuses on mini-batch training

Our work processes at least an order of magnitude more vertices than DistDGL

Weak Scaling

Kronecker [1]



GAT - This Work
GAT - DistDGL
VA - This Work
VA - DistDGL
AGNN - This Work
AGNN - DistDGL

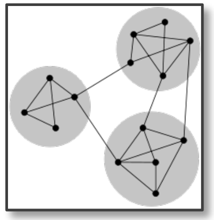
p = 0,1%

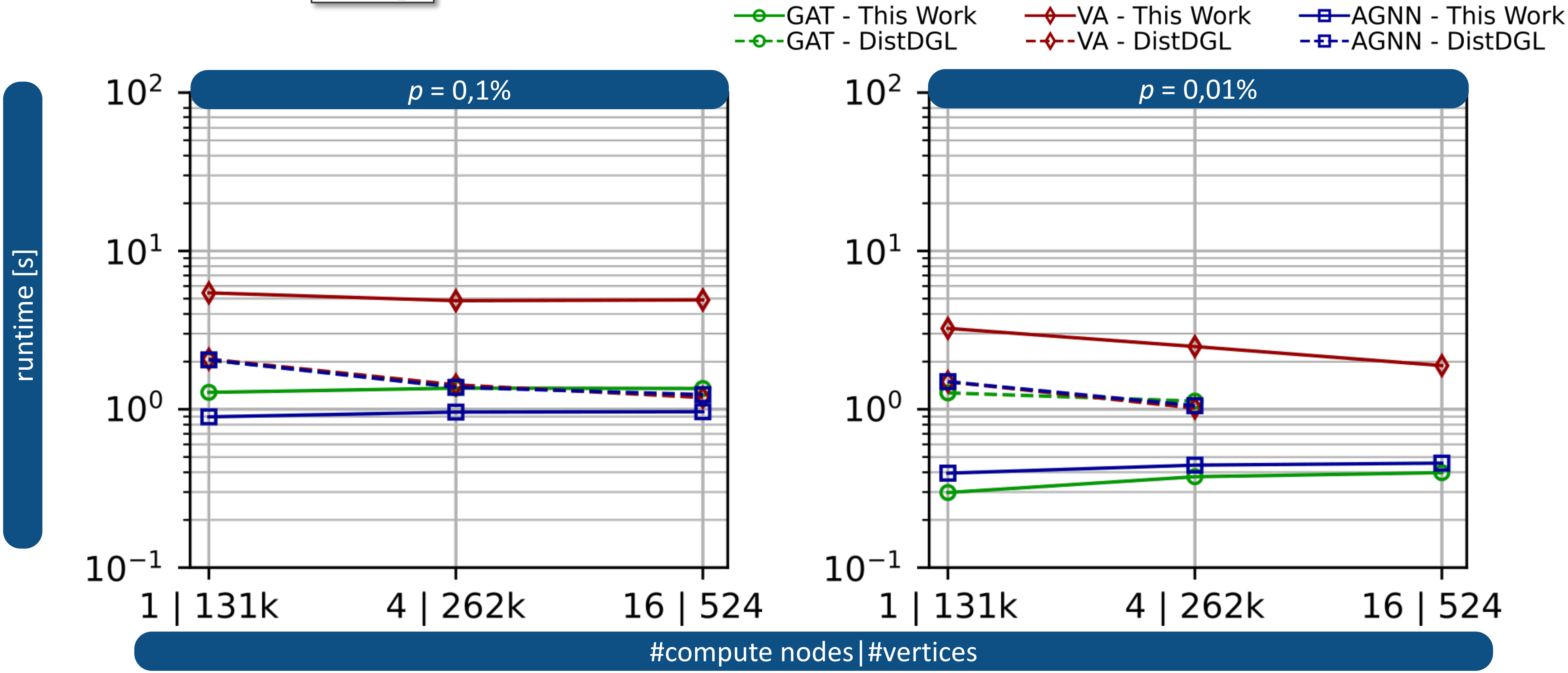
p = 0,01%

[1] J. Leskovec et al. Kronecker Graphs: An Approach to Modeling Networks. J. Mach. Learn. Research. 2010.

Weak Scaling

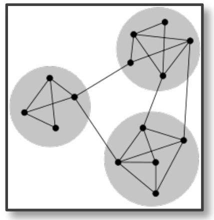
Kronecker [1]

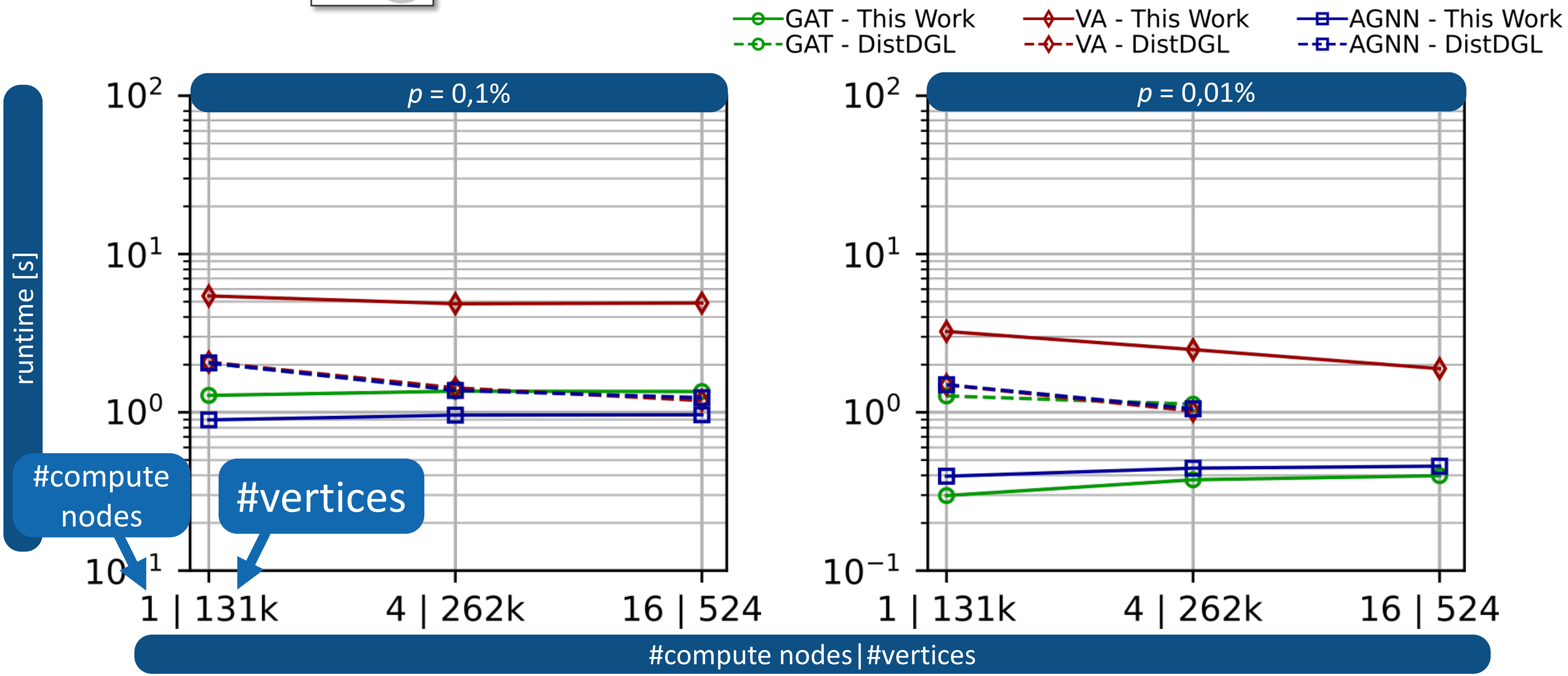




Weak Scaling

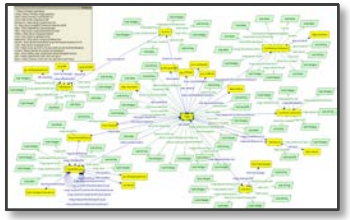
Kronecker [1]





Strong Scaling

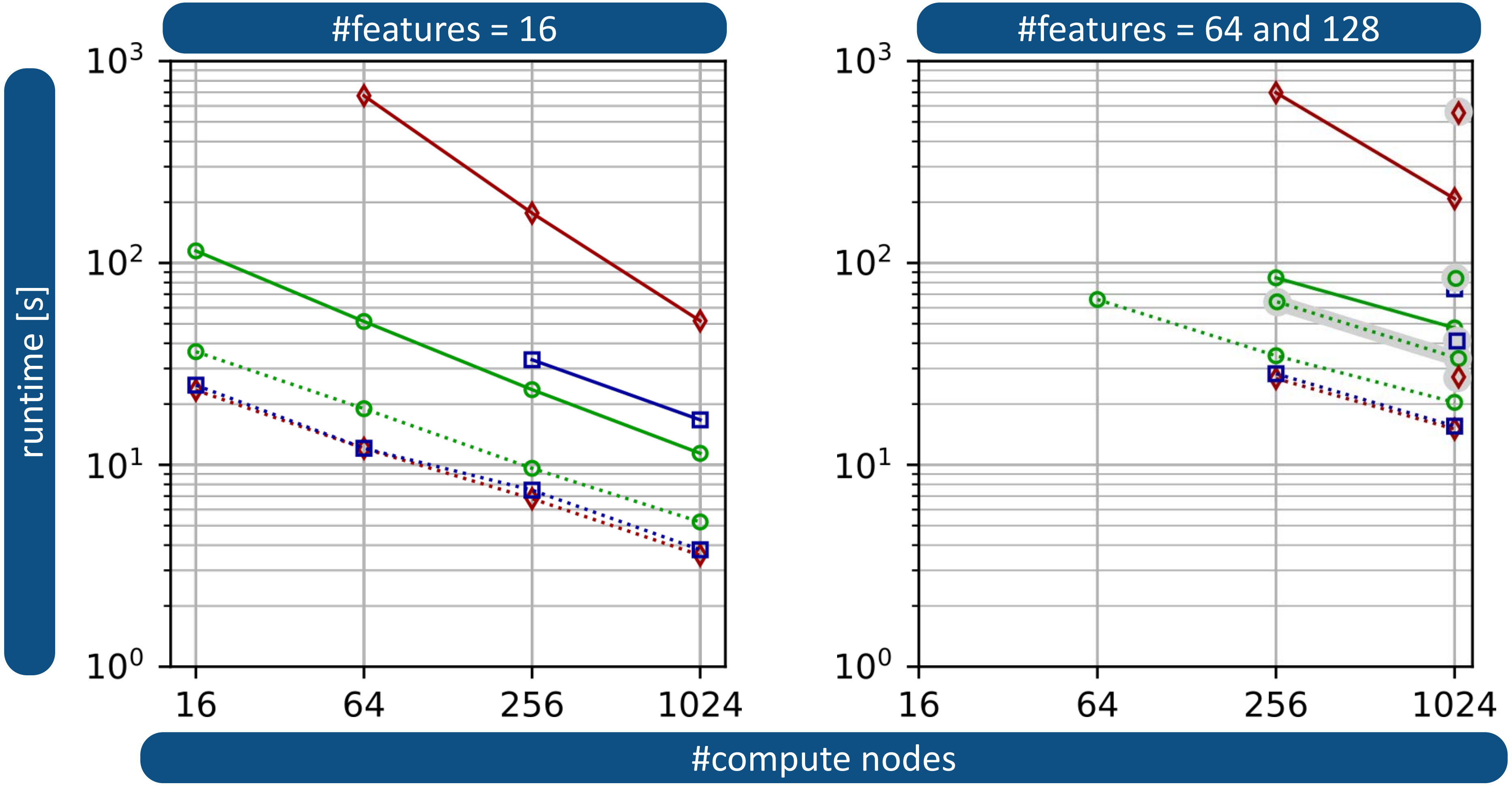
Microsoft Academic Knowledge Graph



Standard real world graph dataset

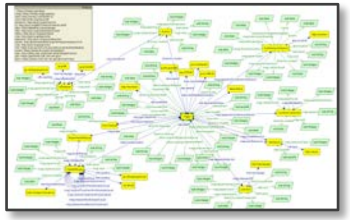
111 million vertices and 3.2 billion edges

GAT - Training
GAT - Inference
VA - Training
VA - Inference
AGNN - Training
AGNN - Inference



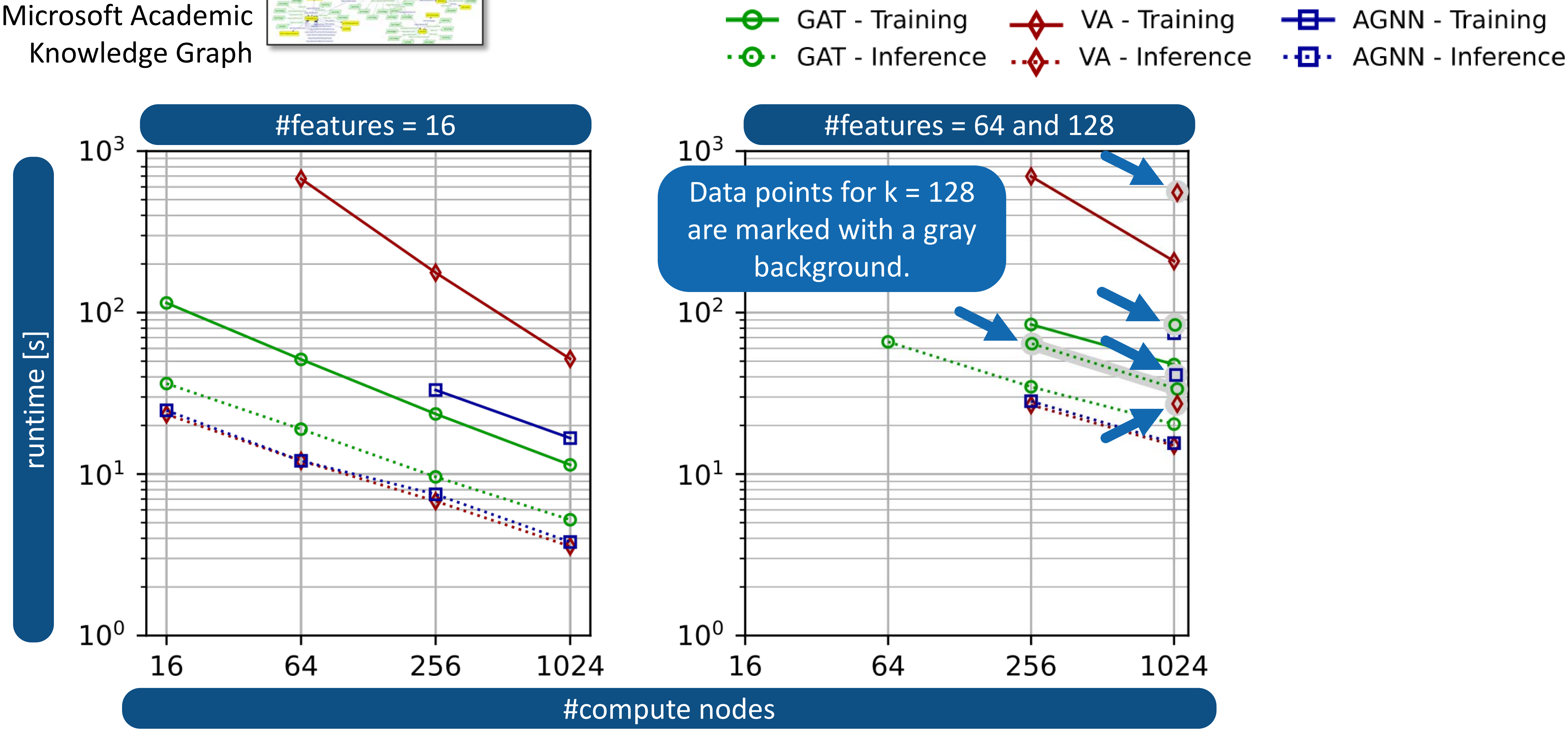
Strong Scaling

Microsoft Academic Knowledge Graph



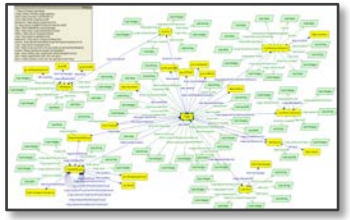
Standard real world graph dataset

111 million vertices and 3.2 billion edges



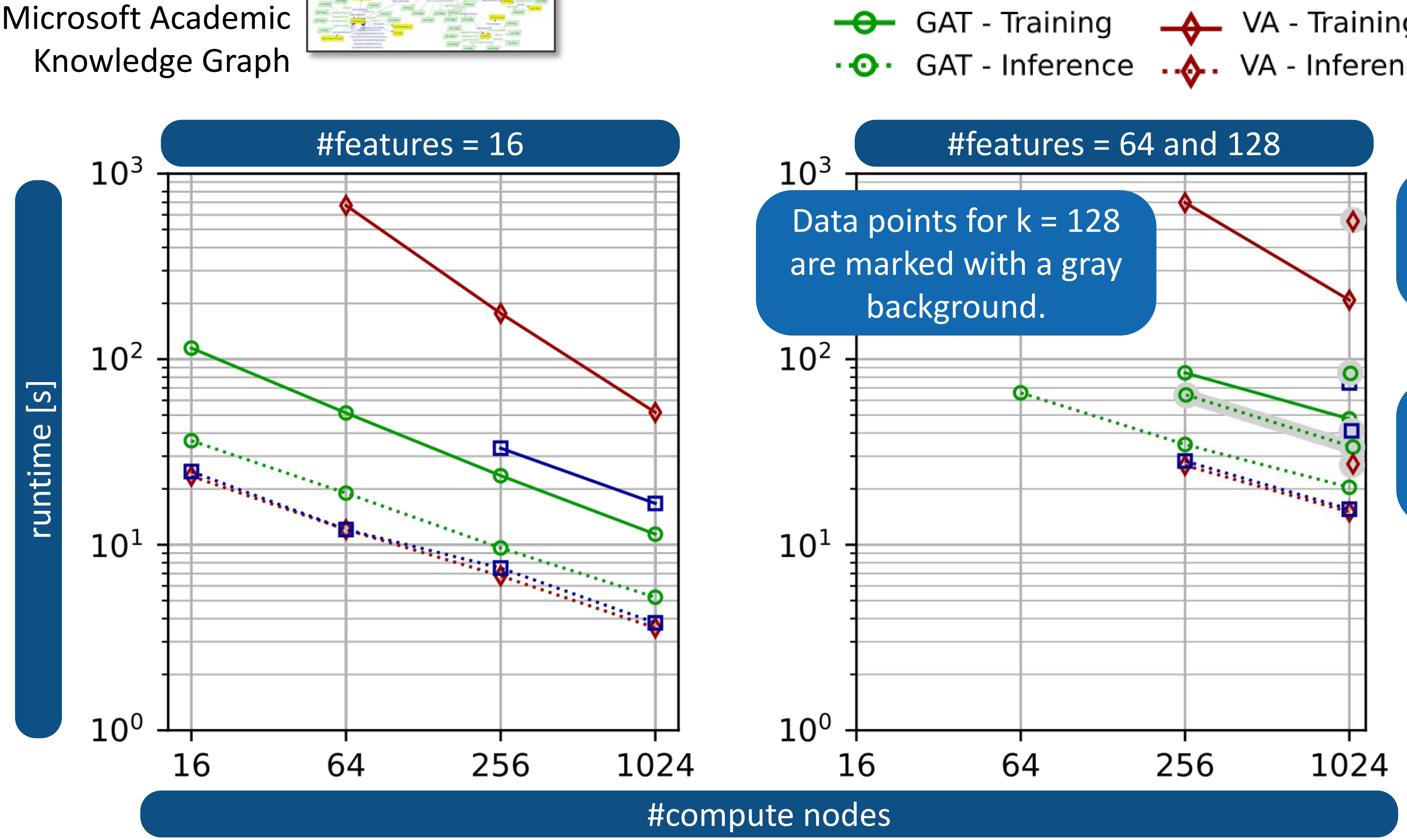
Strong Scaling

Microsoft Academic Knowledge Graph



Standard real world graph dataset

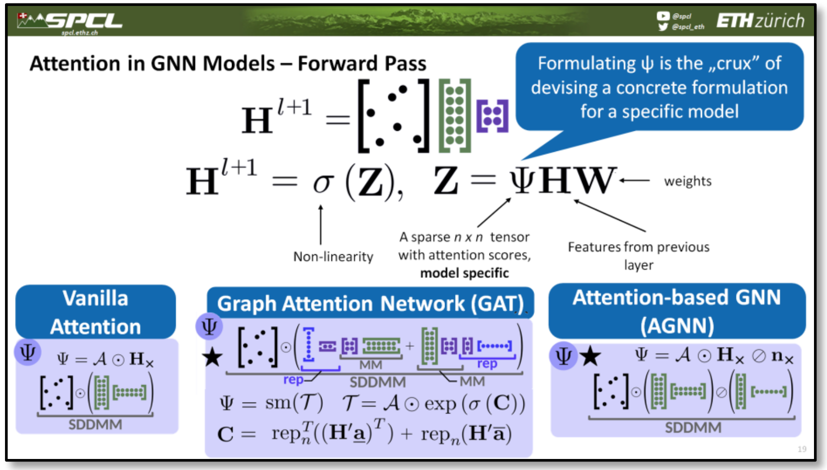
111 million vertices and 3.2 billion edges

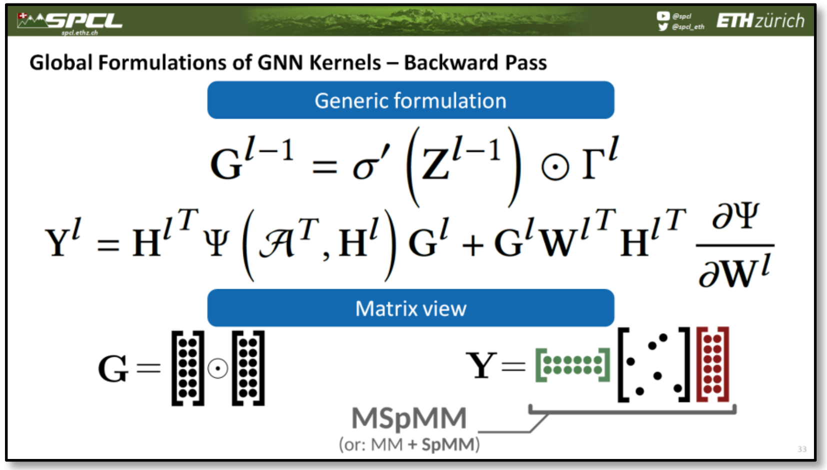


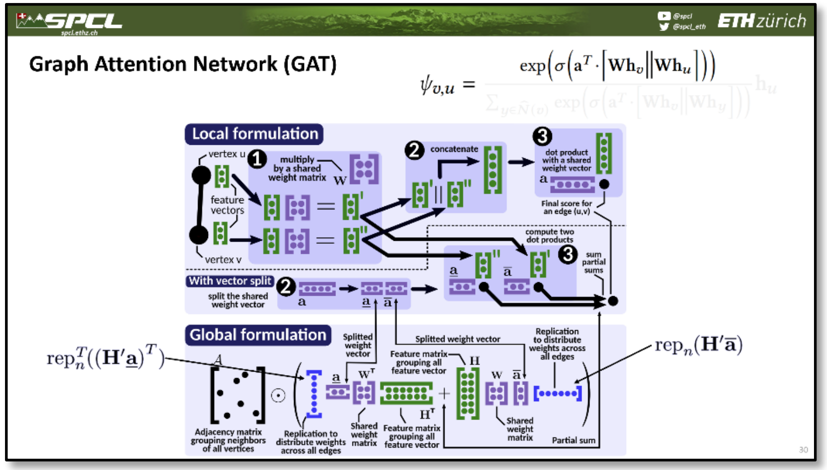
For k = 64, all models except GAT require 256 compute nodes.

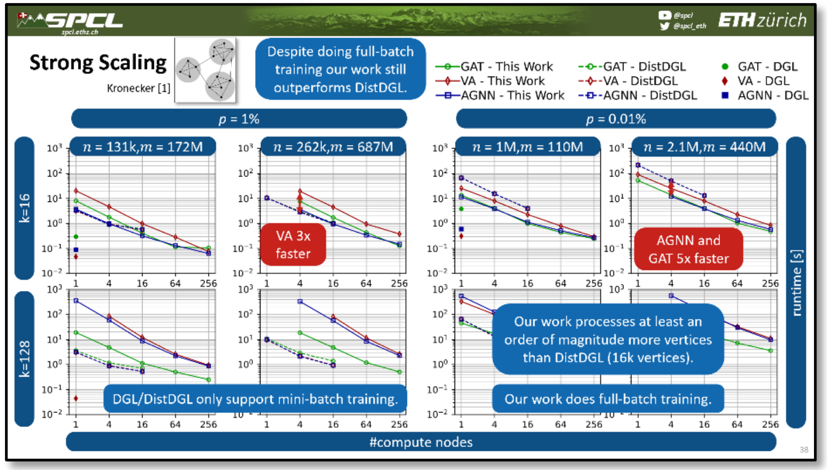
For k = 128, all models except GAT require 1,024 compute nodes.

Conclusions









More of SPCL's research:

- youtube.com/@spcl — 175+ Talks
- twitter.com/spcl_eth — 1.4K+ Followers
- github.com/spcl — 2K+ Stars

… or spcl.ethz.ch

Artifacts Evaluated Functional V1.1 — Artifacts Available V1.1 — Results Reproduced V1.1