



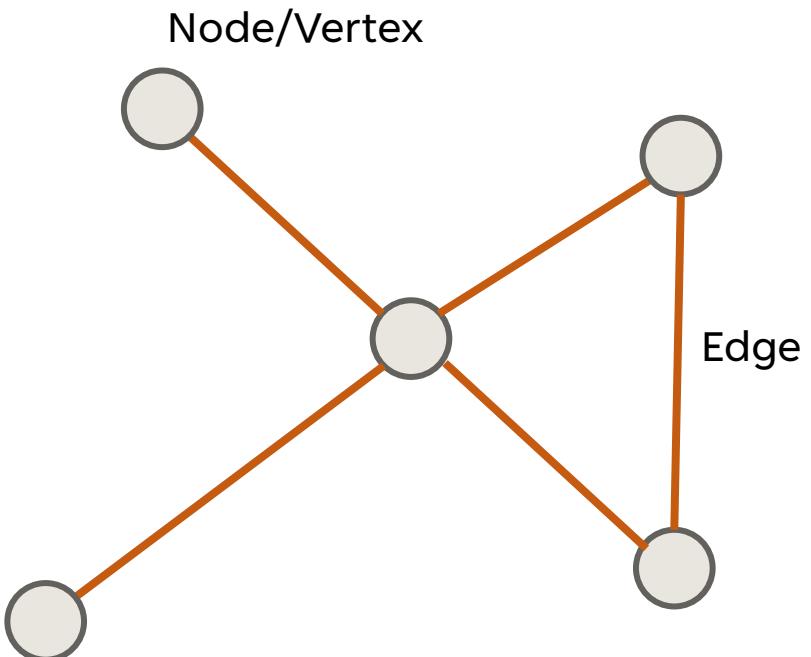
INTRODUCTION TO GRAPH NEURAL NETWORKS

Keith Butler

BEFORE WE START

Let's open up the exercise notebook in colab and put the dependencies loading in the background.

WHAT IS A GRAPH?

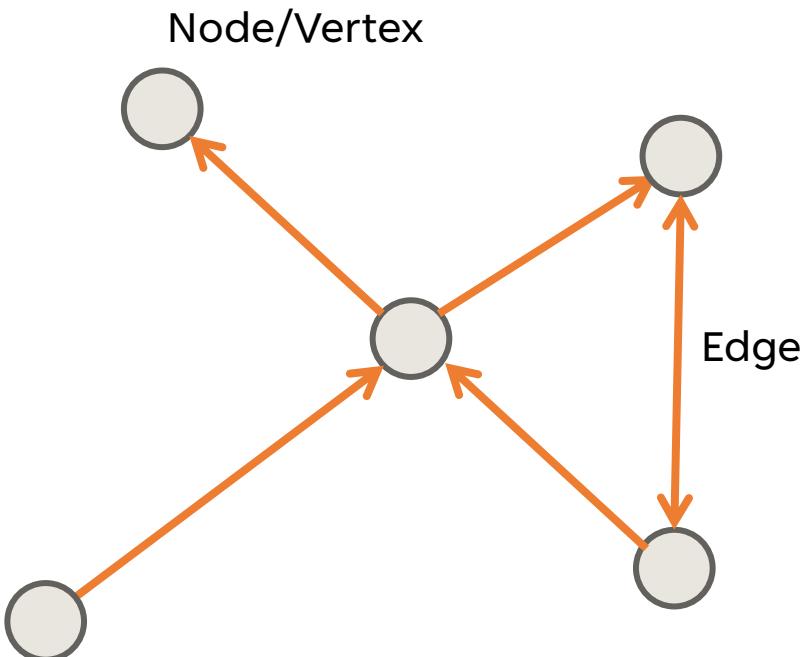


Graphs encode **relations** between **entities**.

Nodes have information about the entity.

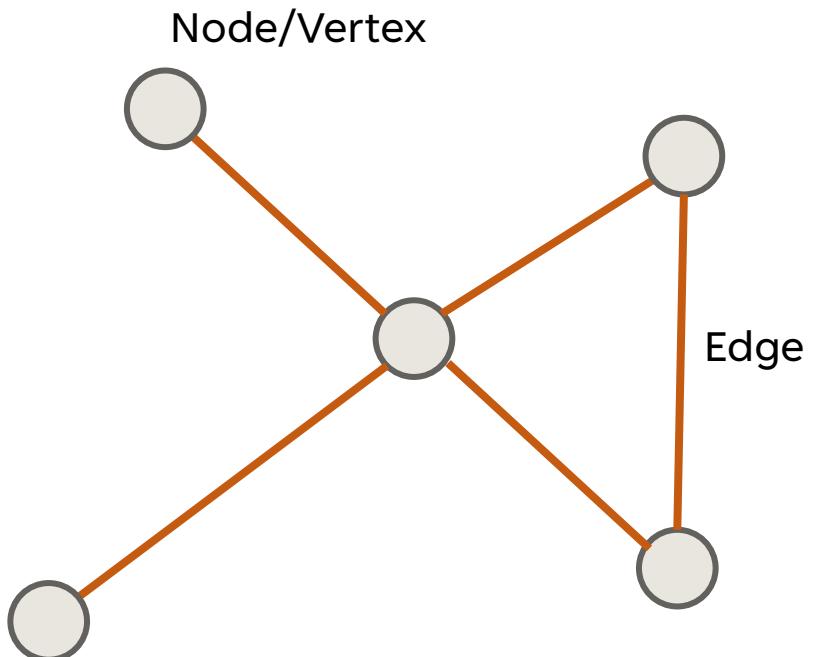
Edges connect nodes.

WHAT IS A GRAPH?



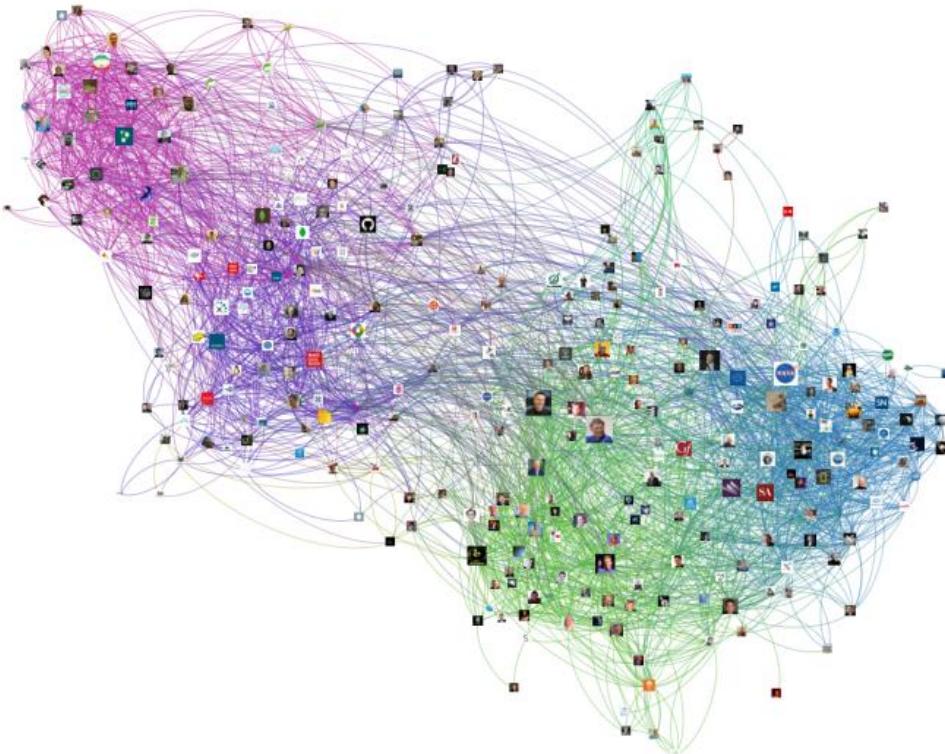
Edges can be **directed** or **undirected**.
Meaning that **information** can flow in either or both directions.

WHAT IS A GRAPH?

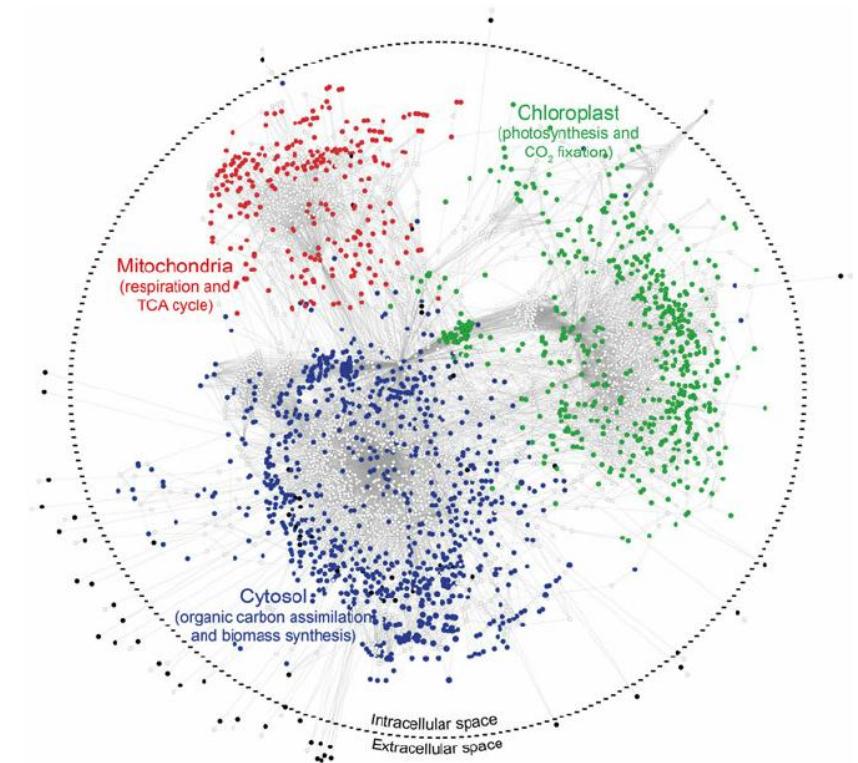


Information is stored in both nodes and edges.
Information is stored as embeddings.

WHERE DO WE FIND/USE GRAPHS?

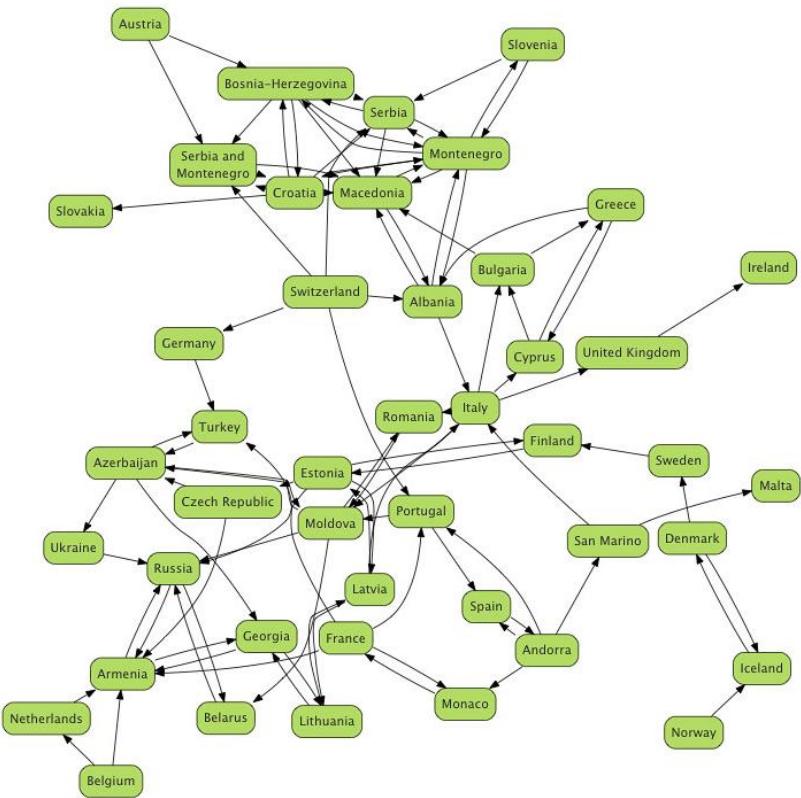


Social networks. >1B nodes; >10B edges

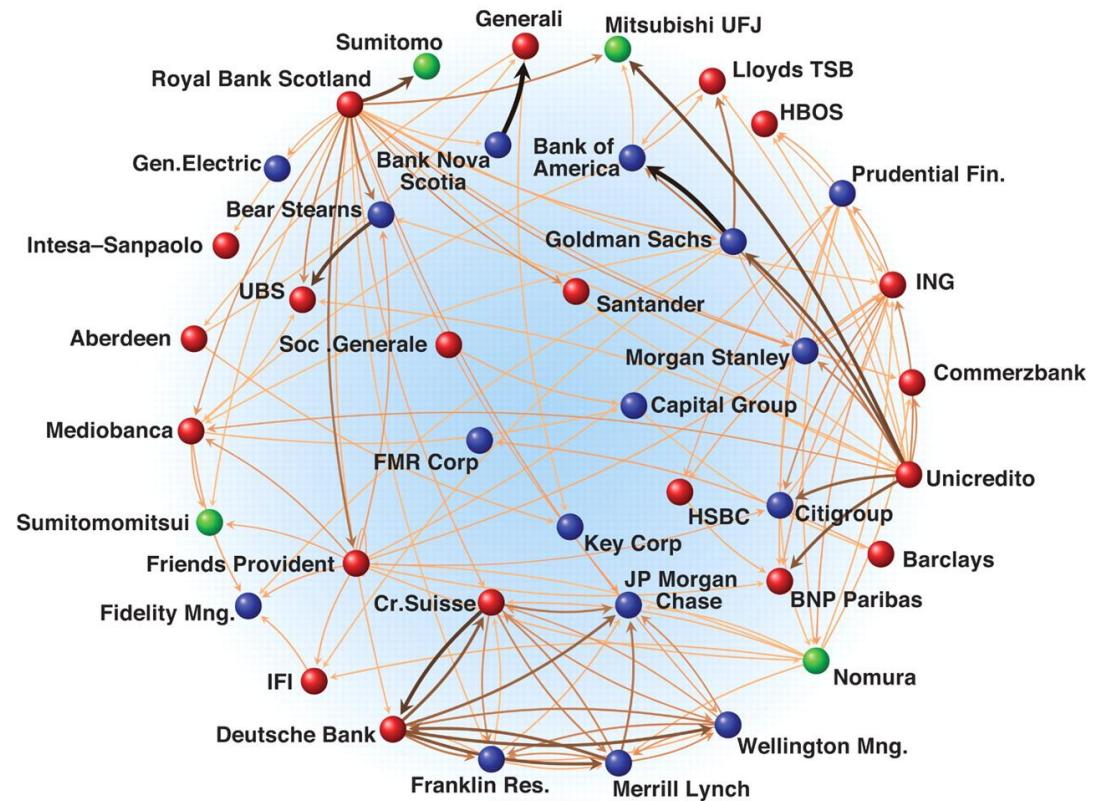


Biological systems

WHERE DO WE FIND/USE GRAPHS?



Eurovision song contest

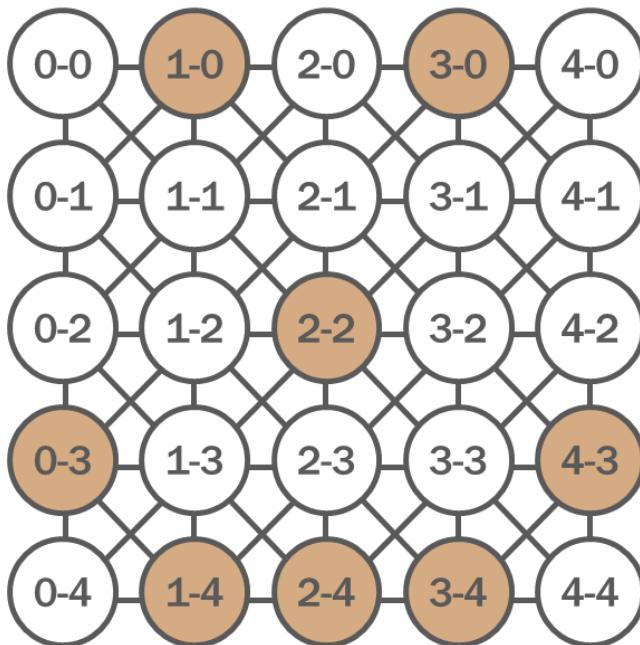


Economics

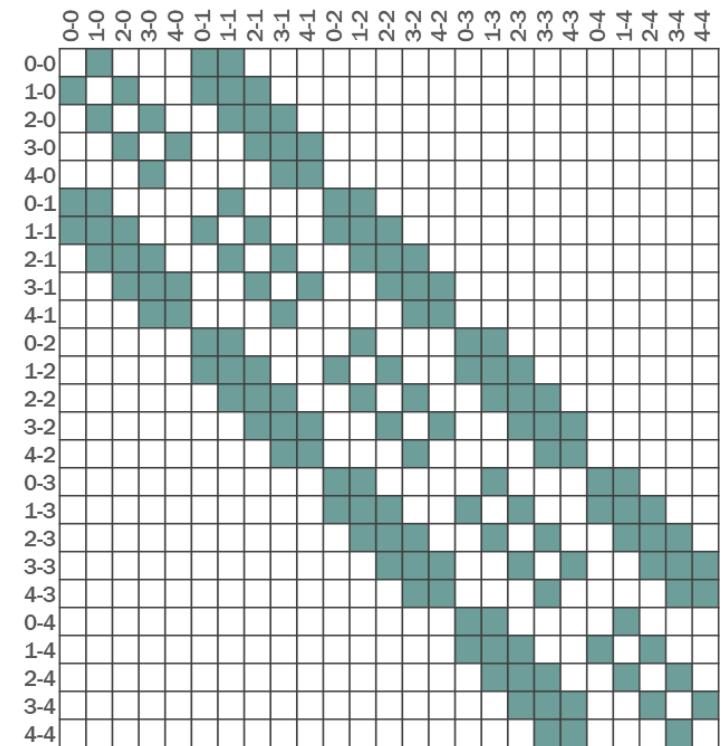
AN IMAGE IS A GRAPH WITH REGULAR STRUCTURE

0-0	1-0	2-0	3-0	4-0
0-1	1-1	2-1	3-1	4-1
0-2	1-2	2-2	3-2	4-2
0-3	1-3	2-3	3-3	4-3
0-4	1-4	2-4	3-4	4-4

Image pixels

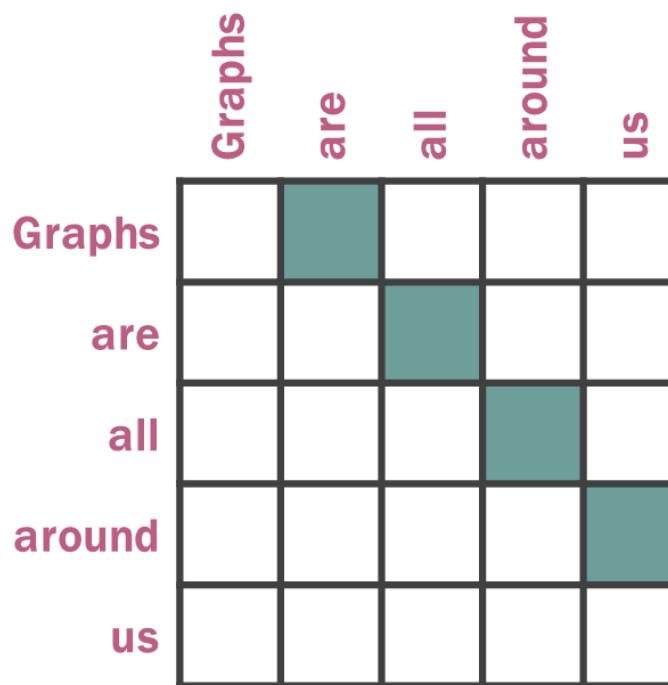


Graph structure

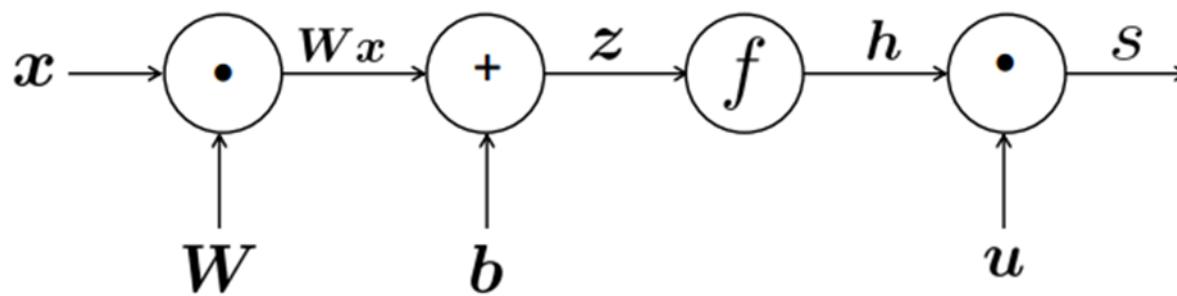


Adjacency matrix

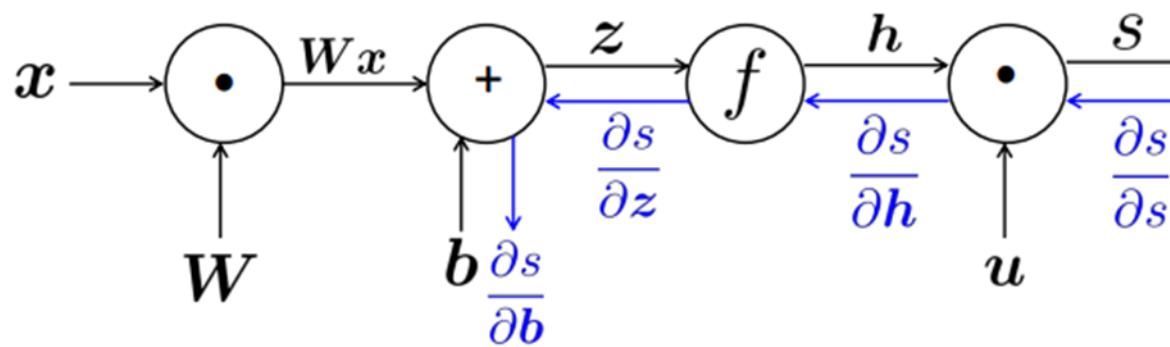
A SENTENCE CAN BE A DIRECTED GRAPH



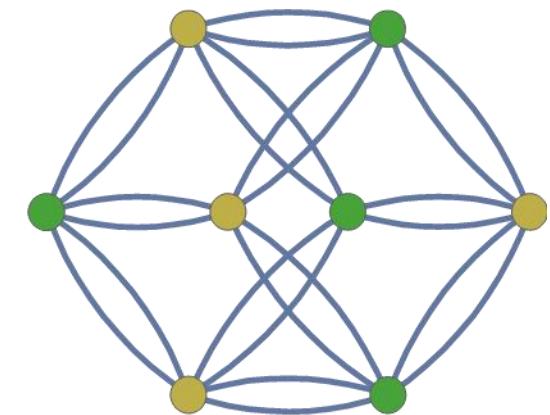
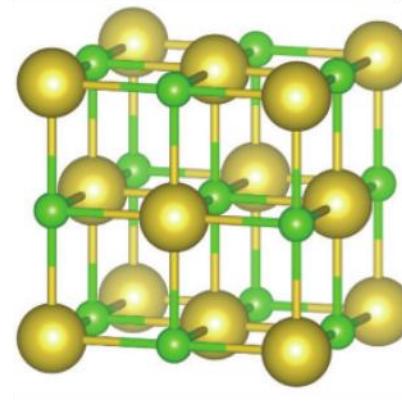
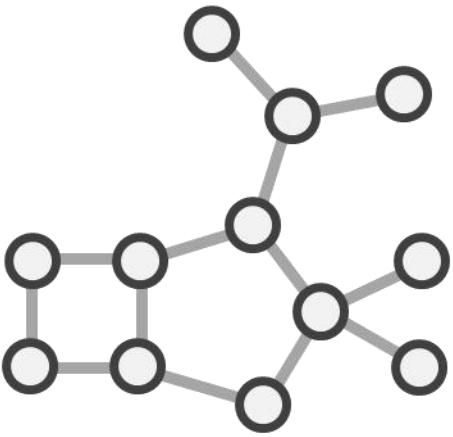
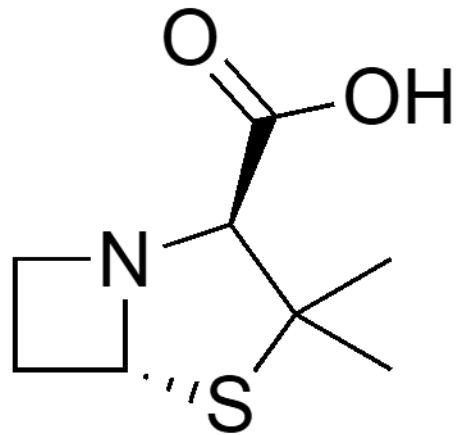
A NEURAL NETWORK IS A GRAPH



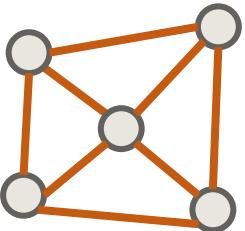
$$\begin{aligned}s &= \mathbf{u}^T \mathbf{h} \\ \mathbf{h} &= f(\mathbf{z}) \\ \mathbf{z} &= \mathbf{W}\mathbf{x} + \mathbf{b} \\ \mathbf{x} &\quad \text{(input)}\end{aligned}$$



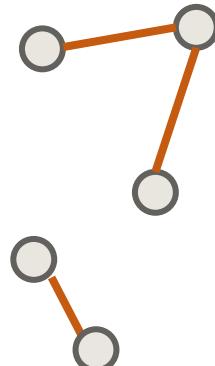
GRAPHS ARE A NATURAL REPRESENTATION FOR CHEMISTRY



ALL GRAPHS ARE NOT ALIKE



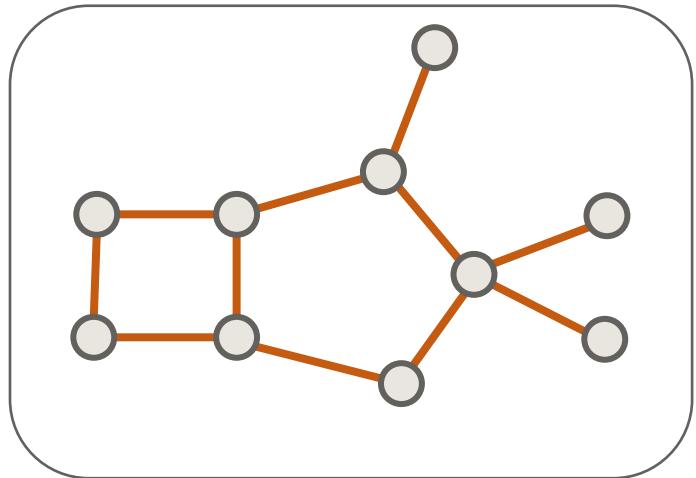
Fully connected



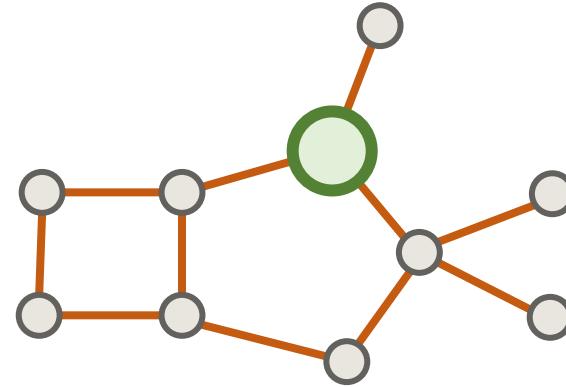
Sparse

Dataset	Graphs	Nodes	Edges
Fully con.	1	5	20
Sparse	2	<4	<3
Wikipedia	1	12M	378M
qm9	134k	<9	<26
Cora	1	23k	91k

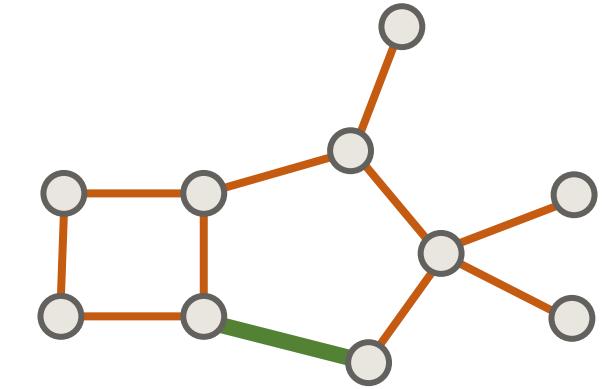
TYPES OF PROPERTIES CALCULATED ON GRAPHS



Graph level e.g. total energy

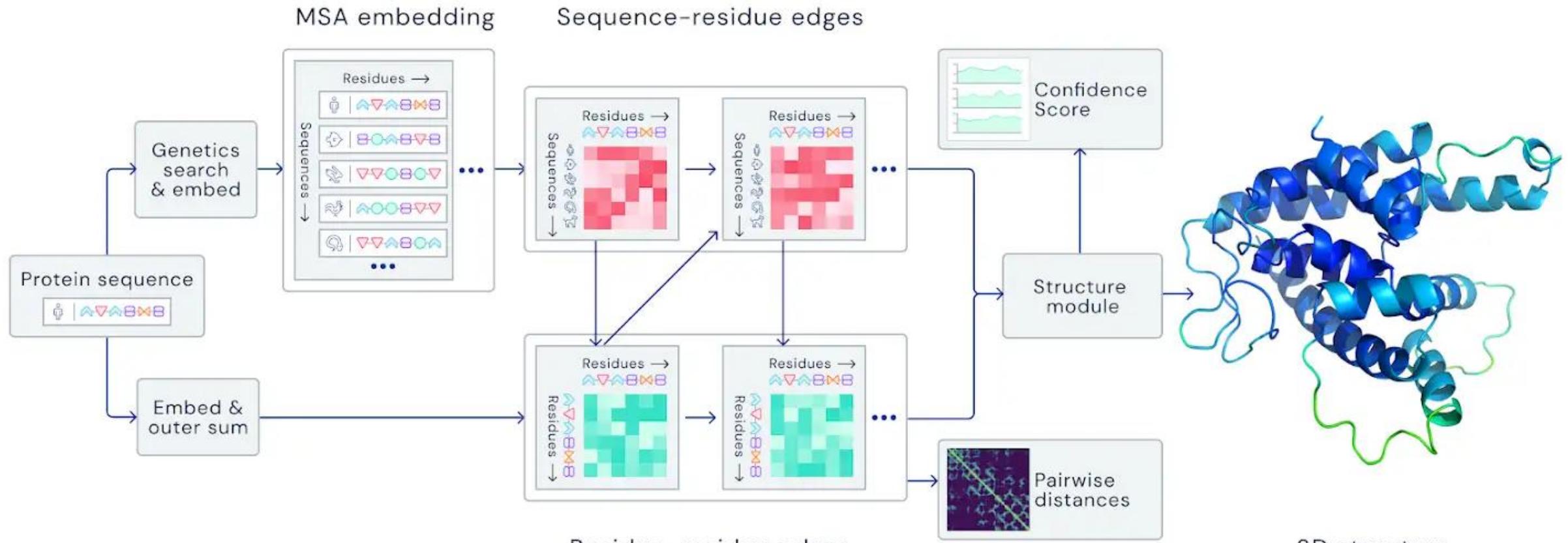


Node level e.g. forces



Edge level e.g. bond order

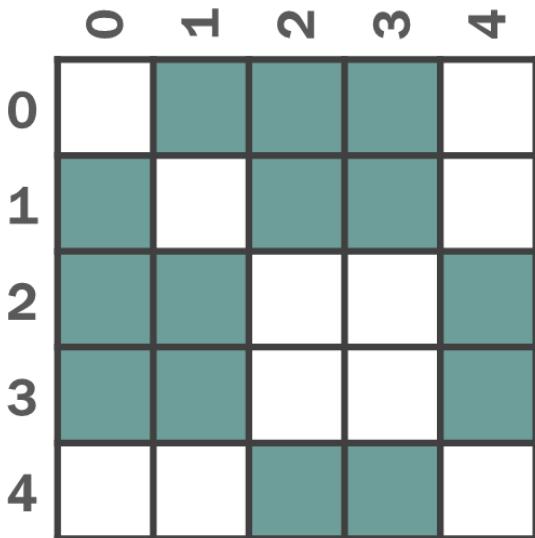
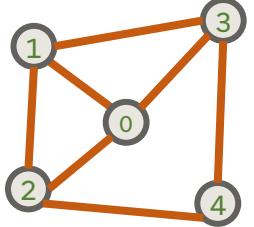
GNNS JUST HELPED WIN A NOBEL PRIZE



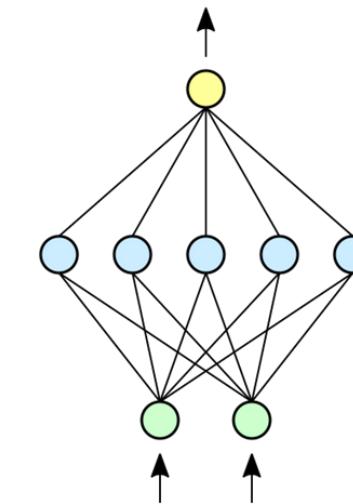
Represent the protein as a graph of amino acids

INCLUDING GRAPHS IN DEEP LEARNING

Could directly use the adjacency matrix



	0	1	2	3	4
0	0	1	1	1	1
1	1	0	1	1	0
2	1	1	0	0	1
3	1	1	0	0	1
4	0	0	1	1	0



Issues: variable size and order dependency

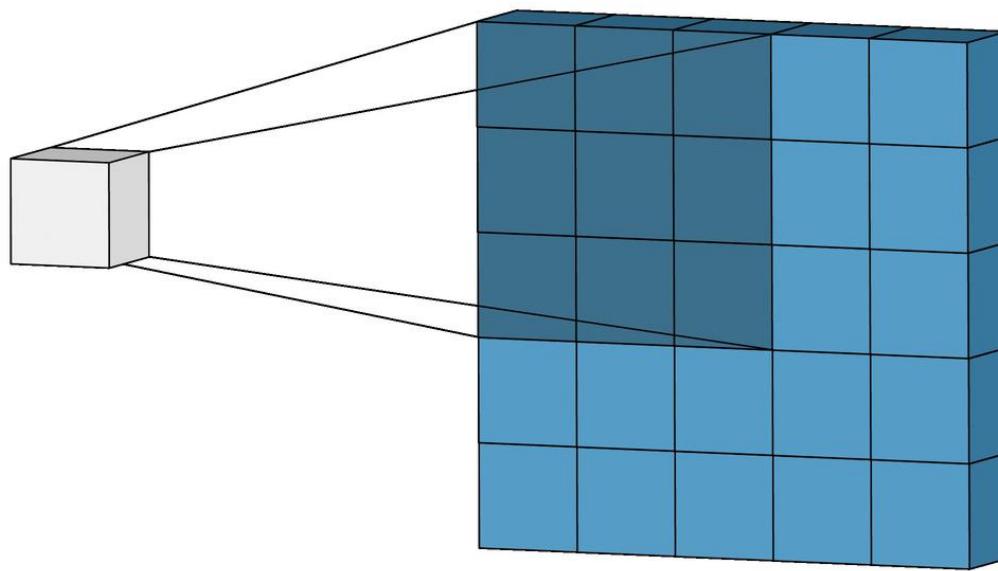
CONVOLUTIONS FOR GRAPHS

A **convolutional neural network** (CNN) filter transforms and combines **information** from neighbouring pixels in an image

0	-1	0
-1	4	-1
0	-1	0

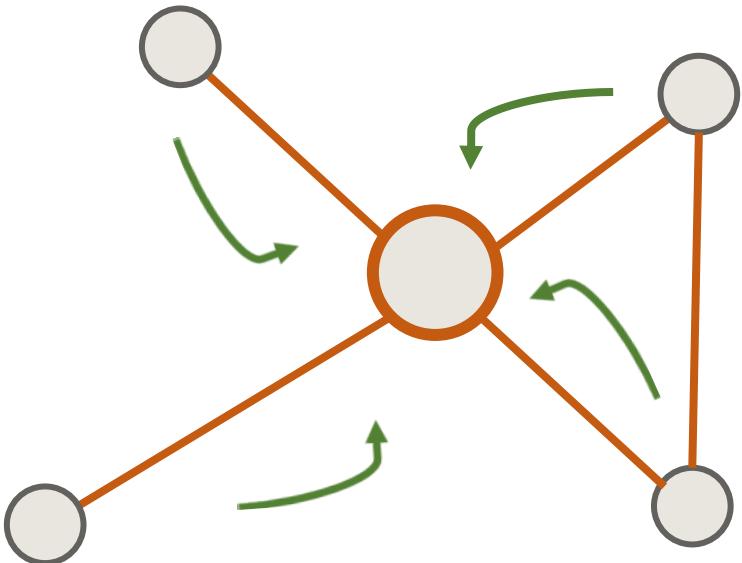
Convolution filter

*learned during training to extract
higher level features e.g., edges*



CONVOLUTIONS FOR GRAPHS

Images can be seen as a regular graph;
can we **extend the concept of convolutions?**



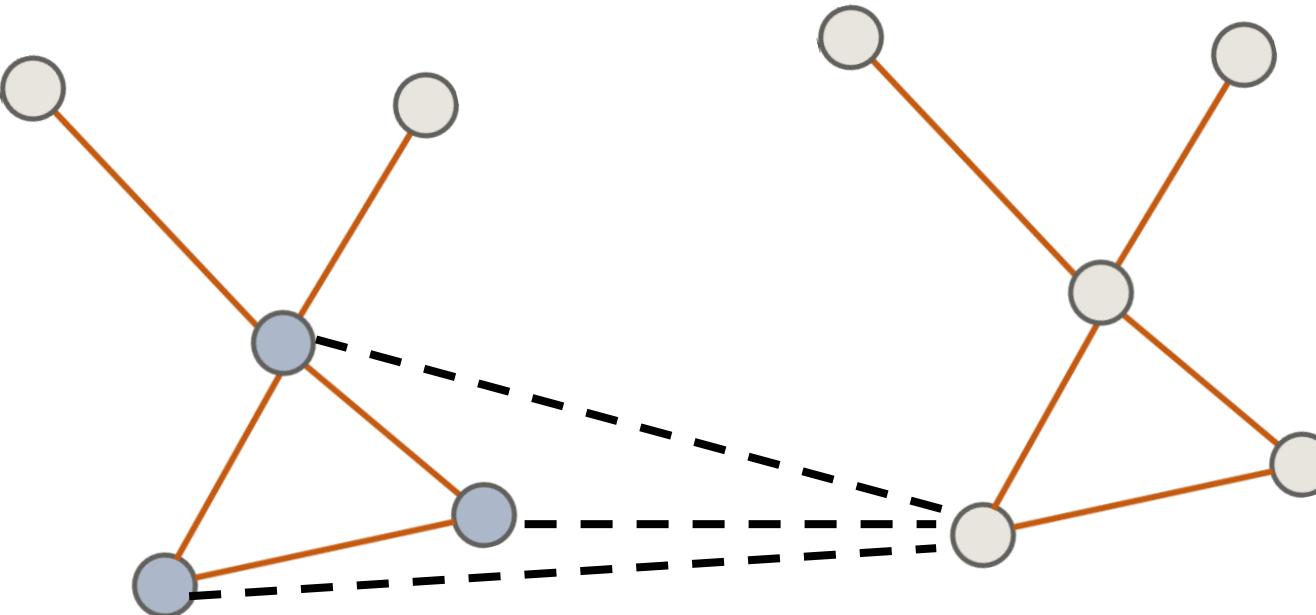
Convolution from **neighbour nodes**



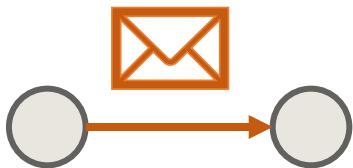
Convolution to **centre nodes**

CONVOLUTIONS FOR GRAPHS

By iterating over the entire graph each node receives information from its neighbours



WHERE DO NEURAL NETWORKS COME IN?



Message passing
What information flows from one node to the next

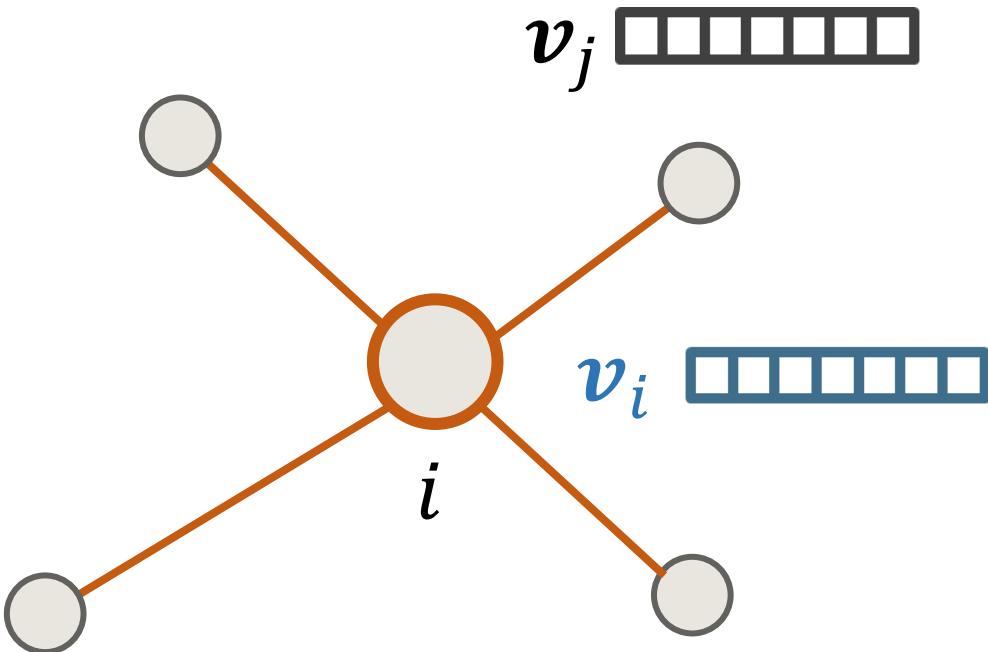


Message pooling
How neighbouring information is added together



Node updates
How the received information changes the node

HOW THE MESSAGE GETS PASSED



$$\mathbf{m}_i = \bigoplus_{j \in \mathcal{N}(i)} M_t(\mathbf{v}_i, \mathbf{v}_j)$$

Message passing function

Message pooling function

$$\mathbf{v}'_i = U_t(\mathbf{v}_i, \mathbf{m}_i)$$

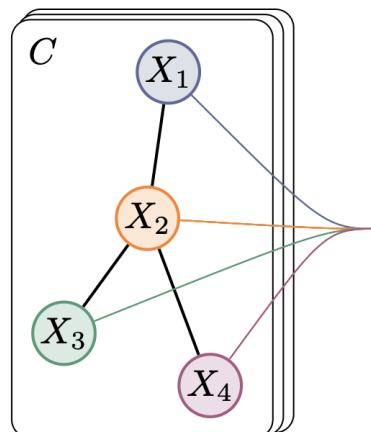
Node update function

THE FIRST GRAPH CONVOLUTIONAL NETWORKS

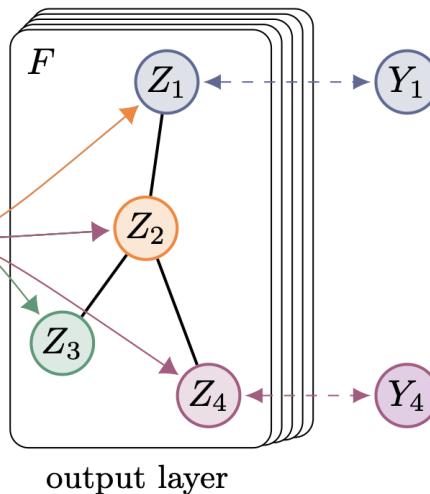
SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

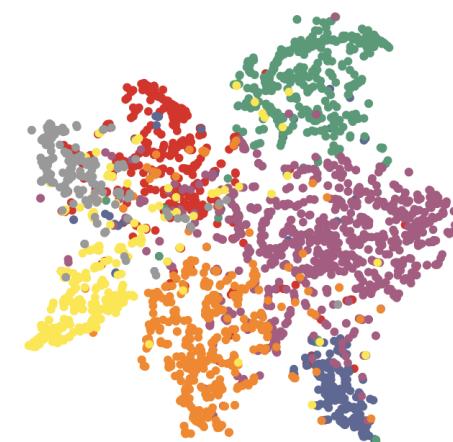
Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl



hidden
layers



(a) Graph Convolutional Network



(b) Hidden layer activations

IMPLEMENTATION OF A GNN

Message

$$\mathbf{v}_j$$

No processing, node vector

Message pooling

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{v}_j}{|\mathcal{N}(i)|}$$

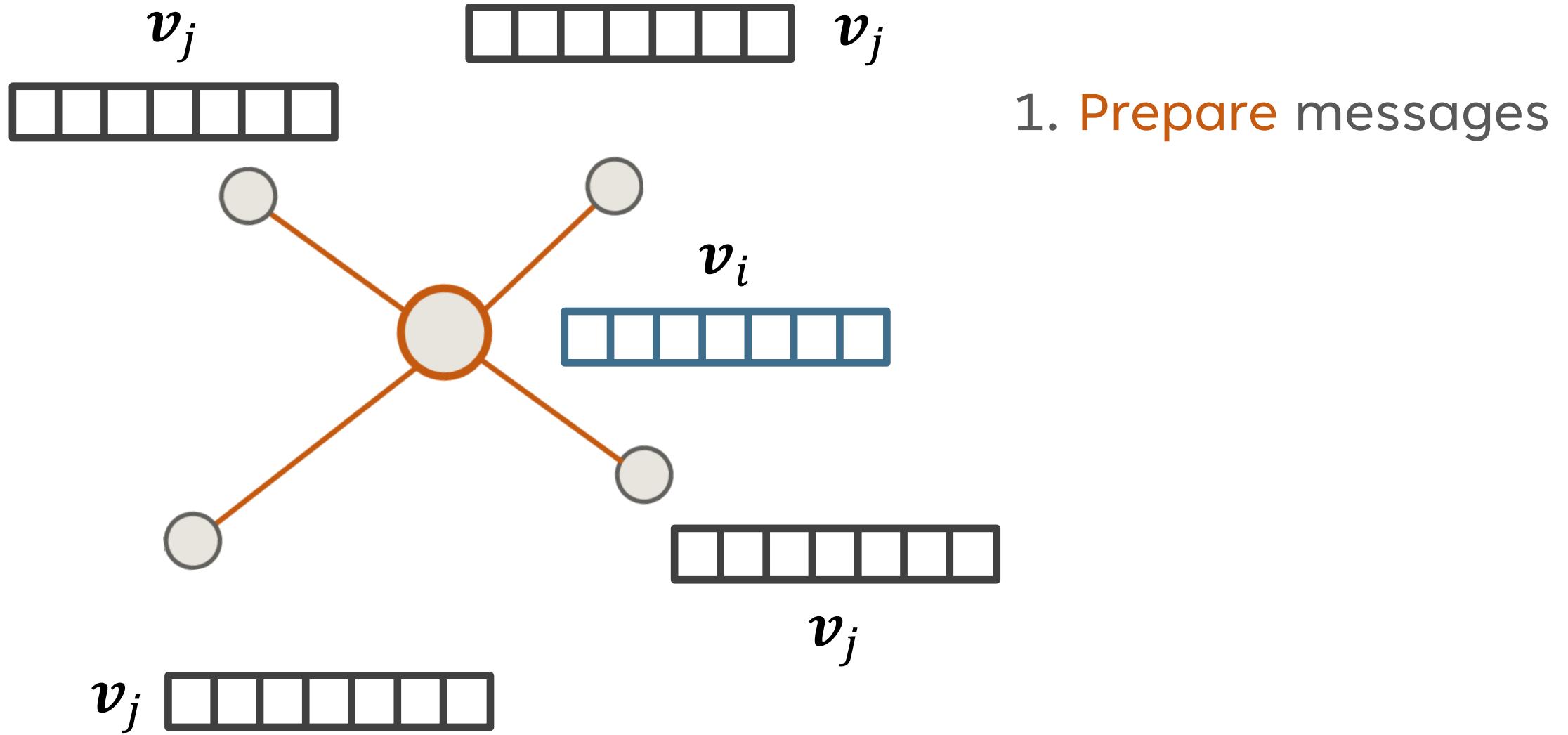
Mean pool across all neighbours

Node update

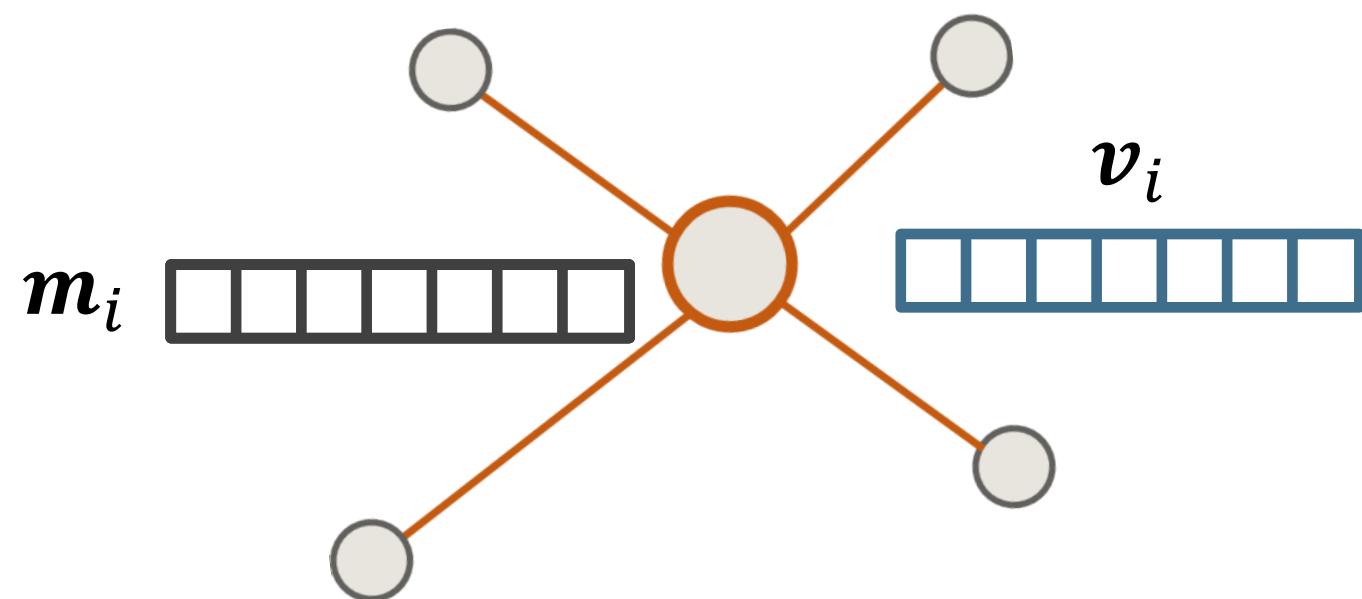
$$\mathbf{v}'_i = \sigma(\mathbf{W}\mathbf{m}_i + \mathbf{B}\mathbf{v}_i)$$

Parameterized learnable function - MLP

VISUALISATION OF A GNN CONVOLUTION



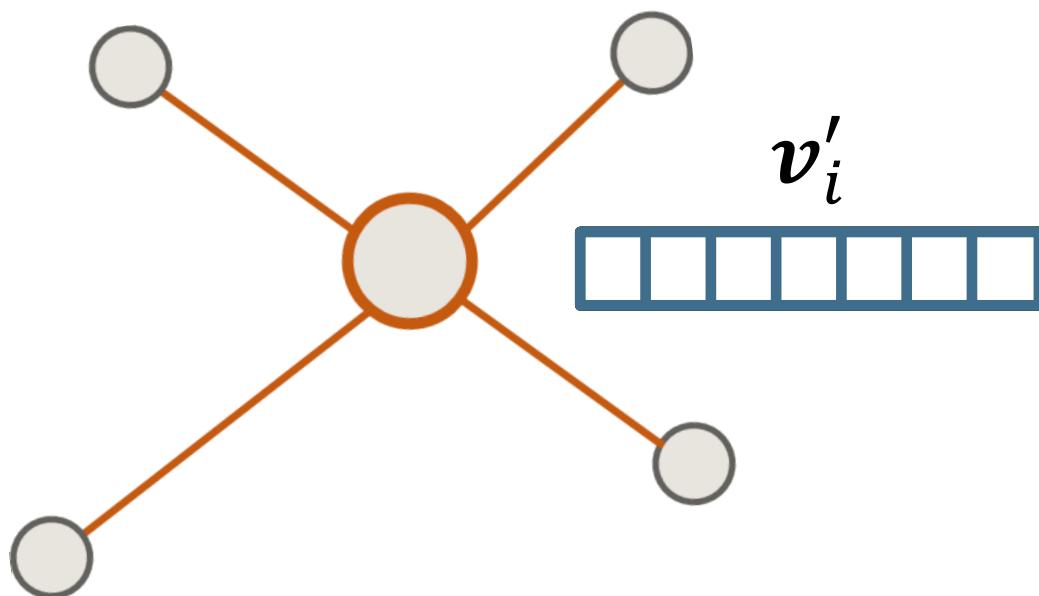
VISUALISATION OF A GNN CONVOLUTION



1. Prepare messages

2. Pool messages

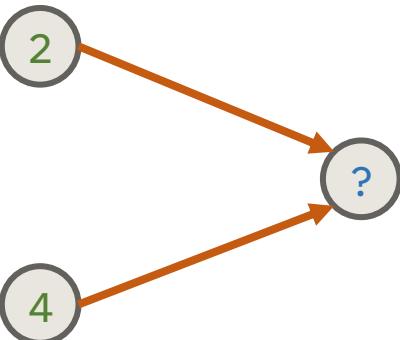
VISUALISATION OF A GNN CONVOLUTION



1. **Prepare messages**
2. **Pool messages**
3. **Update embedding**

PROPERTIES OF THE POOLING FUNCTION

The **pooling** function must be invariant to **node ordering**
and the **number of nodes**



Function	Node value
Max	4
Mean	3
Sum	6

TRAINING A GNN

$$\mathbf{v}'_i = \sigma \left(\mathbf{W} \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{v}_j}{|\mathcal{N}(i)|} + \mathbf{B} \mathbf{v}_i \right)$$

Feed the final node embeddings to a **loss function**

Run an optimiser to train the weight parameters

W and **B** are **shared** across all nodes

EFFICIENCY AND INDUCTIVE CAPABILITY

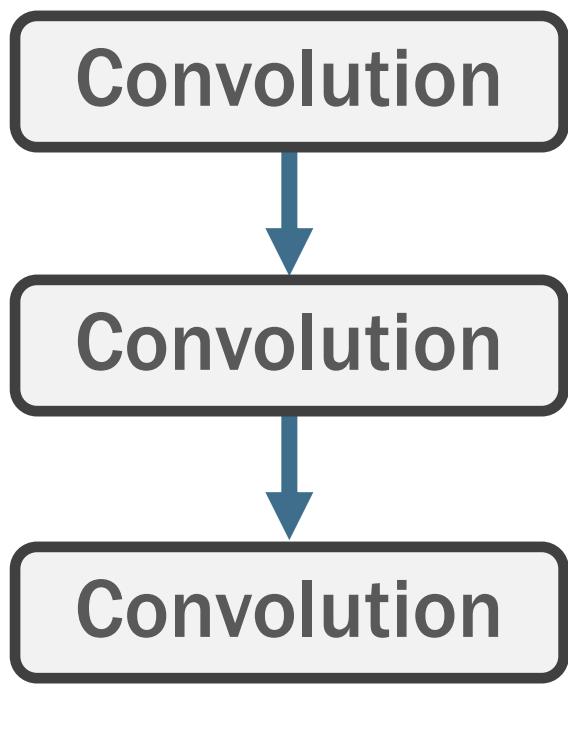
Each node **has its own network** due to its connectivity

Message, pool, and update functions are **shared for all nodes**

Can increase number of nodes **without increasing
the number of parameters**

Can introduce new unseen node structures and just **plug in
the same matrices**

STACKING CONVOLUTIONS

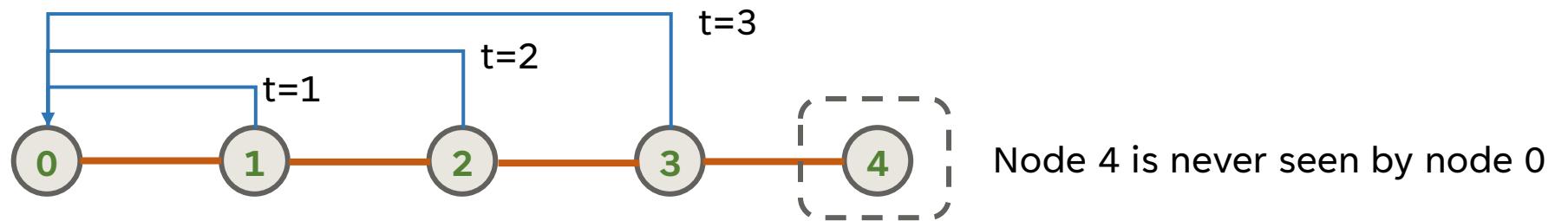


$$v_i^{(t+1)} = \sigma \left(\mathbf{W}^{(t)} \sum_{j \in \mathcal{N}(i)} \frac{v_j^{(t)}}{|\mathcal{N}(i)|} + \mathbf{B}^{(t)} v_i^{(t)} \right)$$

Weights are unique for each layer

THE ADVANTAGE OF MULTIPLE CONVOLUTIONS

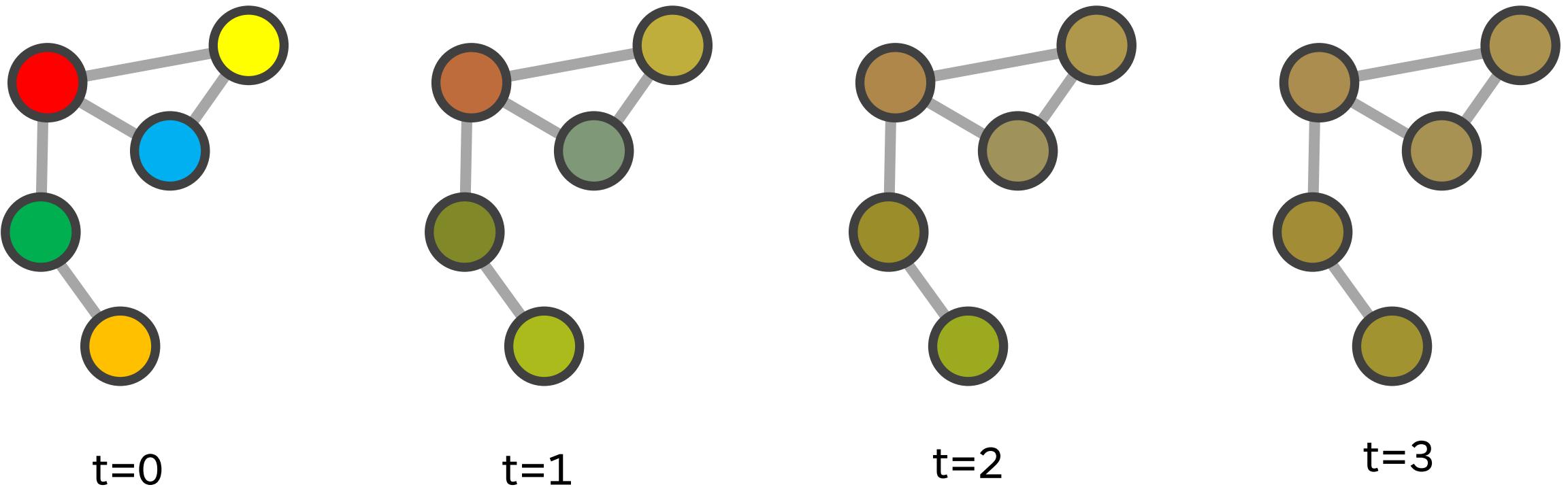
Graphs are **inherently local** – they only get information up to t convolutions away



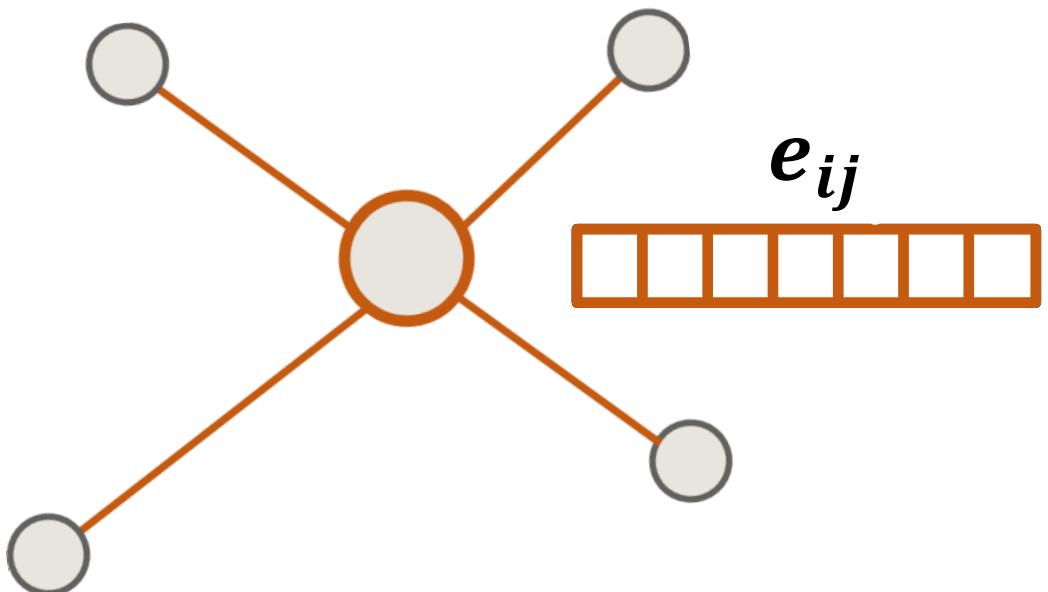
Stacking convolutions increases the **receptive field** of the graph

THE DRAWBACK OF MULTIPLE CONVOLUTIONS

However, too many convolutions causes **over smoothing** –
all node embeddings **converge to the same value**



EDGE EMBEDDINGS



$$\mathbf{m}_i = \bigoplus_{j \in \mathcal{N}(i)} M_t(v_i, v_j, e_{ij})$$

Edge embedding

$$v'_i = U_t(v_i, \mathbf{m}_i)$$

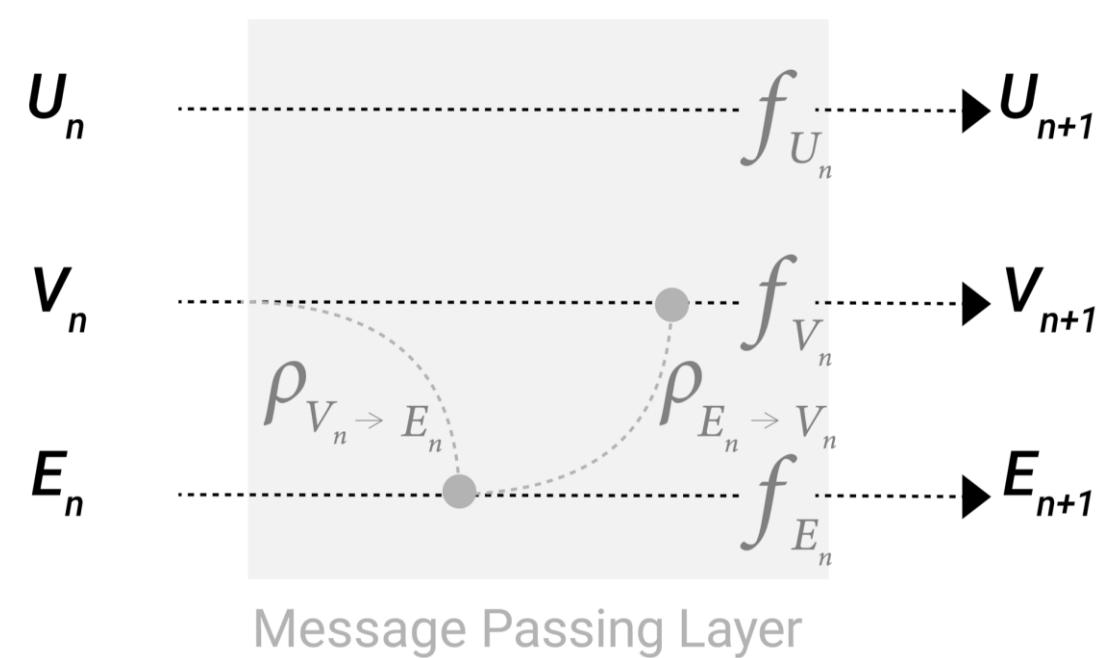
The update function stays
the same

MESSAGE PASSING NETWORKS – SIGNIFICANT FLEXIBILITY

Many options for how to treat edges in the pooling function

Edge embeddings may have different dimensionality to node embeddings

An option is to pool all edges and concatenate them at the end

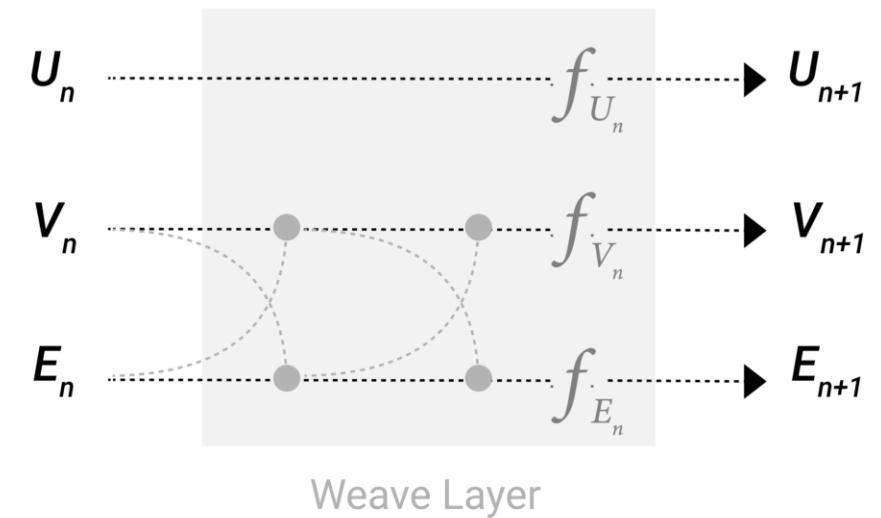
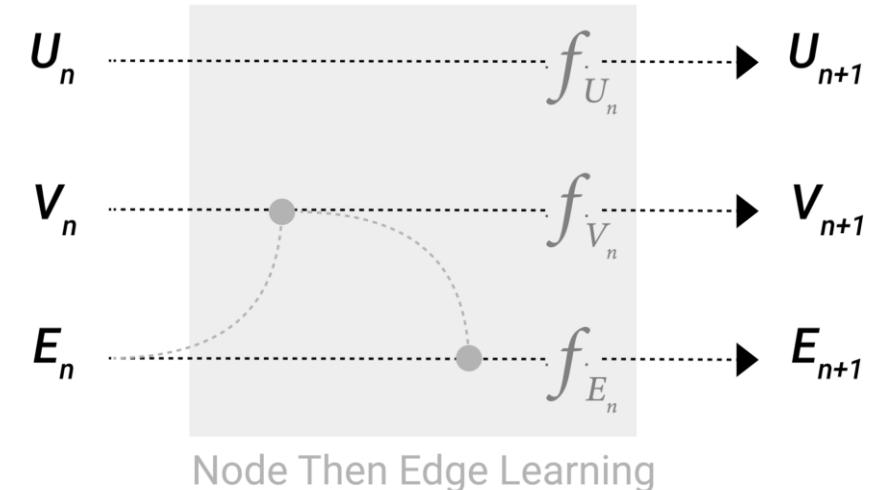


MESSAGE PASSING NETWORKS – SIGNIFICANT FLEXIBILITY

Can update **nodes before edges** or vice versa

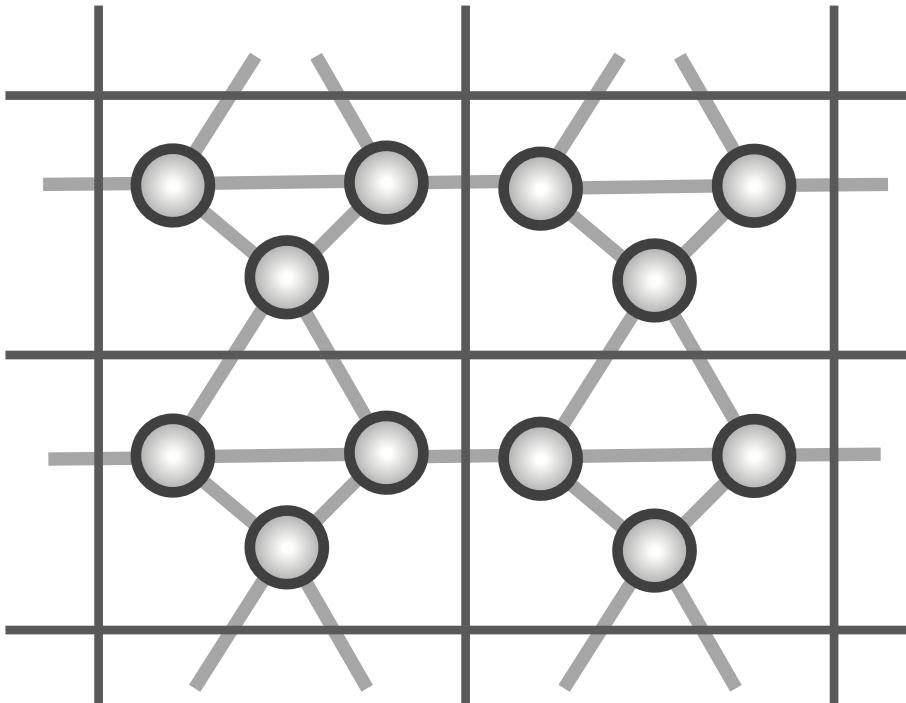
Or have a weave design to **pass messages back and forth**

All flexible design choices in **message passing networks**



CONVOLUTIONAL GRAPH NETWORKS FOR CRYSTALS

Graphs are a natural representation for crystals and but
we have **extra design constraints**

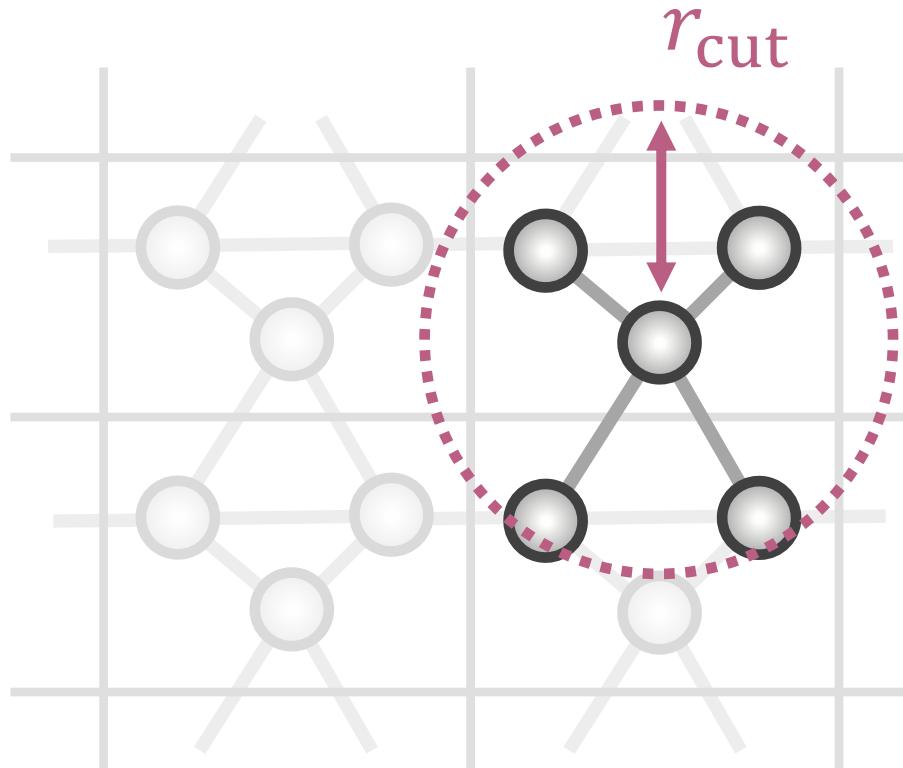


Networks should be **permutation**
and translation invariant

Properties depend on **atom types**
and coordinates not just
connectivity

CONSTRUCTING THE GRAPH FROM A CRYSTAL STRUCTURE

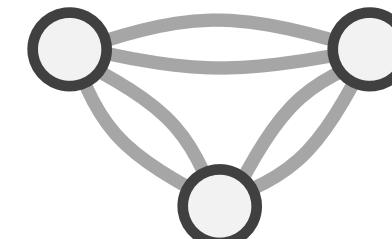
Include all atoms **within a certain cut-off as neighbours**



Must consider periodic boundaries

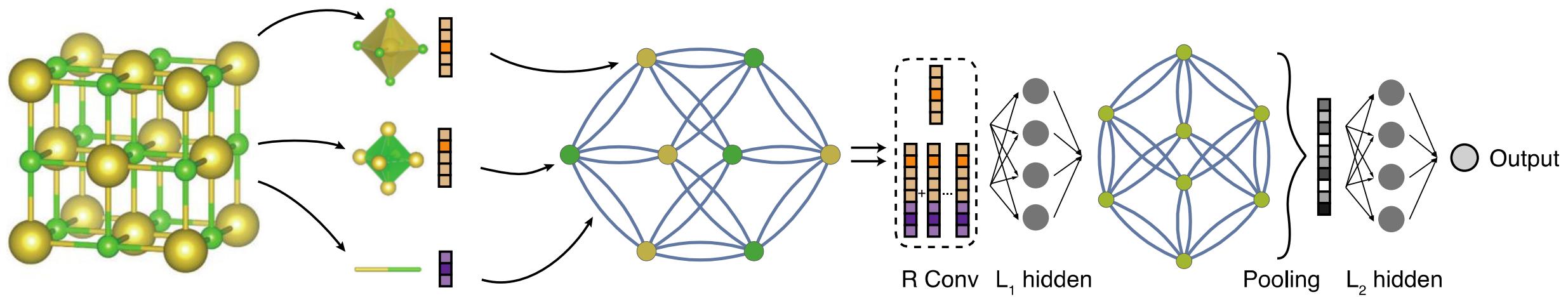
Perform the procedure for **each atom in the unit cell**

Nodes can share **multiple edges to the same neighbour** due to PBC



CRYSTAL GRAPH CONVOLUTIONAL NEURAL NETWORKS (CGCNN)

CGCNN was the first time graph convolutions were applied to crystals

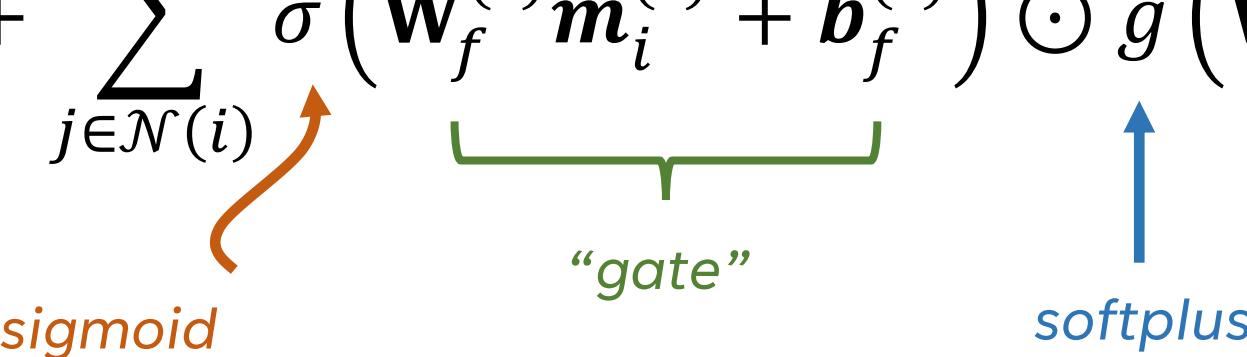


IMPLEMENTATION OF CGCNN

Message function:

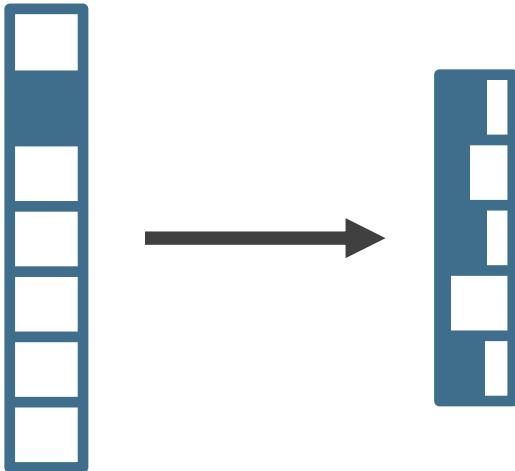
$$\mathbf{m}_i^{(t)} = \mathbf{v}_i^{(t)} \oplus \mathbf{v}_j^{(t)} \oplus \mathbf{e}_{i,j}$$

Update function:

$$\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} \sigma \left(\mathbf{W}_f^{(t)} \mathbf{m}_i^{(t)} + \mathbf{b}_f^{(t)} \right) \odot g \left(\mathbf{W}_s^{(t)} \mathbf{m}_i^{(t)} + \mathbf{b}_s^{(t)} \right)$$


INITIALISATION — NODE AND EDGE EMBEDDINGS

What to do for the **initial node and edge embeddings?**



Nodes

The element type is one-hot encoded (dimension of 119) and passed through an MLP



Edges

The bond distance is projected onto a Gaussian basis (40 basis functions)

READOUT — CALCULATING THE FINAL PREDICTION

CGCNN generates *graph level predictions*, how are these generated from the final node embeddings?

Final pooling of
all nodes

$$\mathbf{u}_c = \sum_{i \in \mathcal{G}} \frac{\mathbf{v}_i^{(T)}}{|\mathcal{G}|}$$

num atoms

SLP
readout

Property predicted

$$E = \sigma(\mathbf{W}_r \mathbf{u}_c + \mathbf{b}_r)$$

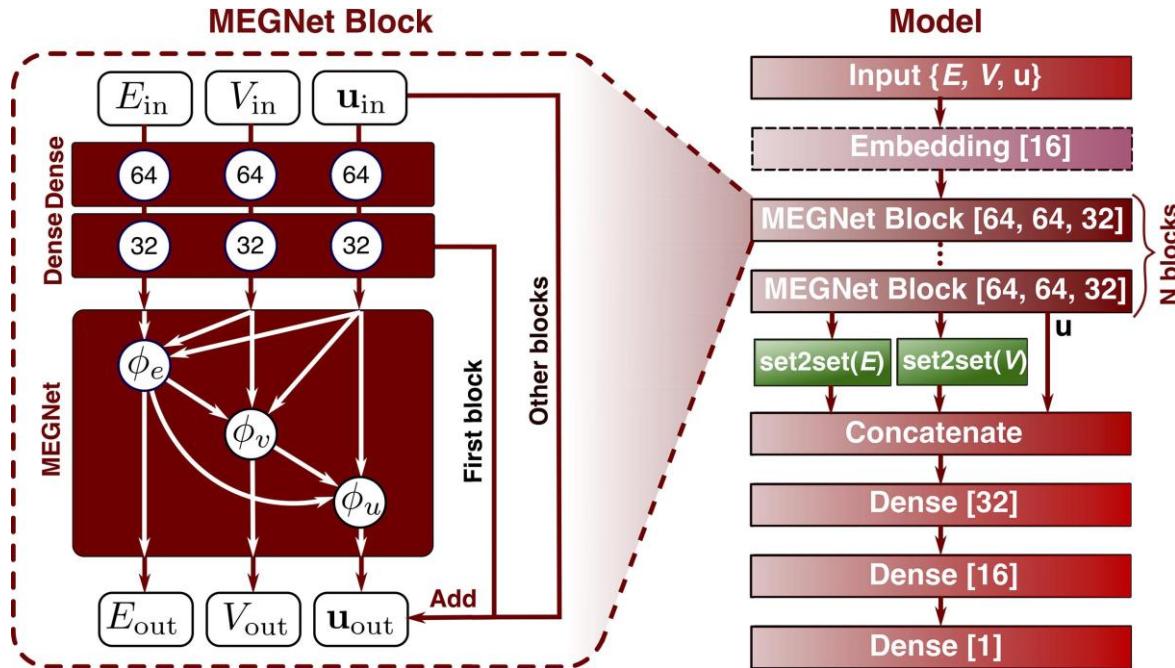
CGCNN PERFORMANCE

CGCNN shows good accuracy for such a simple model but **errors are still too large for reliable science**

Property	# of train data	Unit	MAE _{model}	MAE _{DFT}
Formation energy	28 046	eV/atom	0.039	0.081–0.136 [28]
Absolute energy	28 046	eV/atom	0.072	...
Band gap	16 458	eV	0.388	0.6 [32]
Fermi energy	28 046	eV	0.363	...
Bulk moduli	2041	log(GPa)	0.054	0.050 [13]
Shear moduli	2041	log(GPa)	0.087	0.069 [13]
Poisson ratio	2041	...	0.030	...

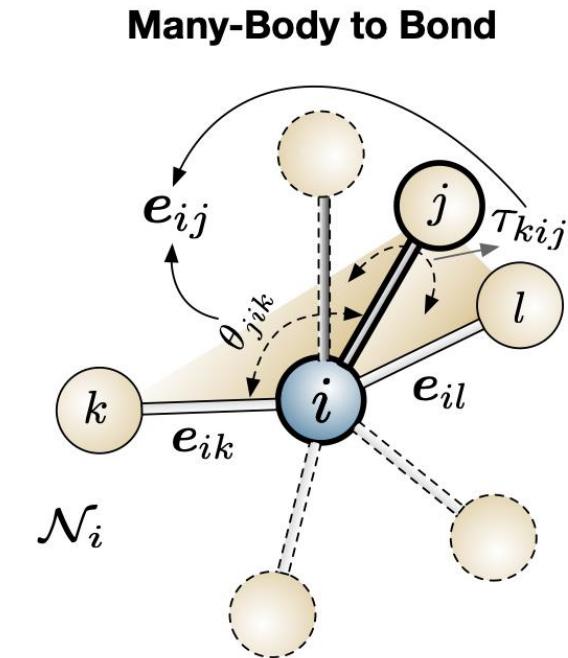
ADVANCED MESSAGE PASSING NETWORKS

CGCNN only uses bond lengths as features. More advanced networks **show improved performance**



MEGNet

Crystal features and set2set pooling



M3GNet

Bond angles and dihedrals

Equivariance vs Invariance

CGCNN only uses bond lengths as features. More advanced networks **show improved performance**

- **Invariant function:**

- Output does *not change* under a transformation

- Example:

- Total energy is invariant under rotation

- **Equivariant function:**

- Output transforms *predictably* with the input

- Example:

- Forces rotate when the molecule rotates

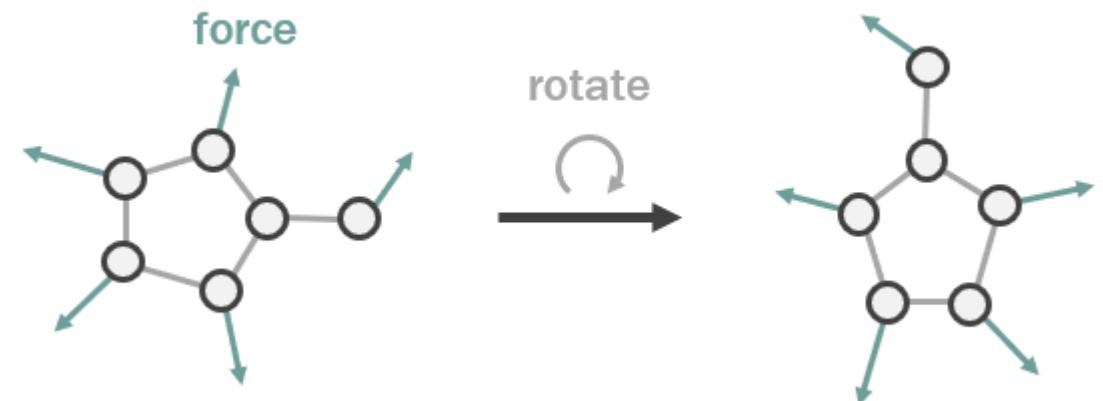
- **Formally:**

- Invariance:

$$f(Tx) = f(x)$$

- Equivariance:

$$f(Tx) = Tf(x)$$



Equivariance vs Invariance

CGCNN only uses bond lengths as features. More advanced networks **show improved performance**

- **Invariant function:**

- Output does *not change* under a transformation

- Example:

- Total energy is invariant under rotation

- **Equivariant function:**

- Output transforms *predictably* with the input

- Example:

- Forces rotate when the molecule rotates

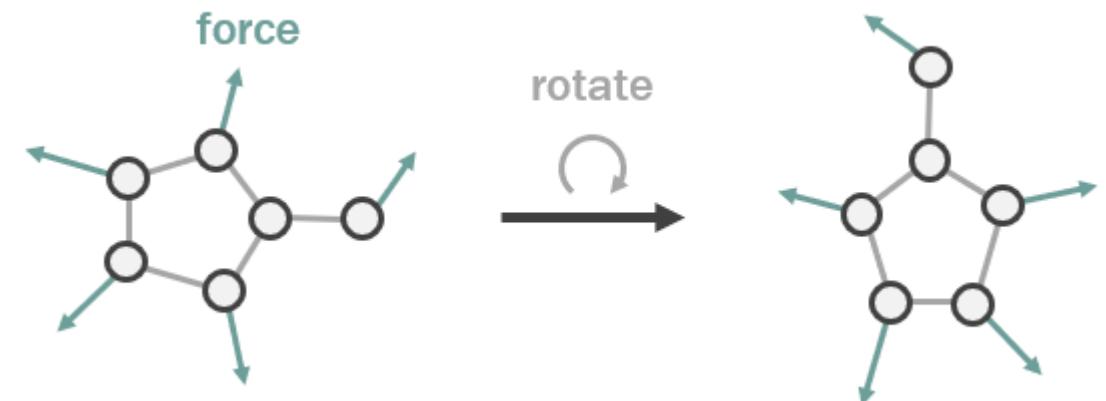
- **Formally:**

- Invariance:

$$f(Tx) = f(x)$$

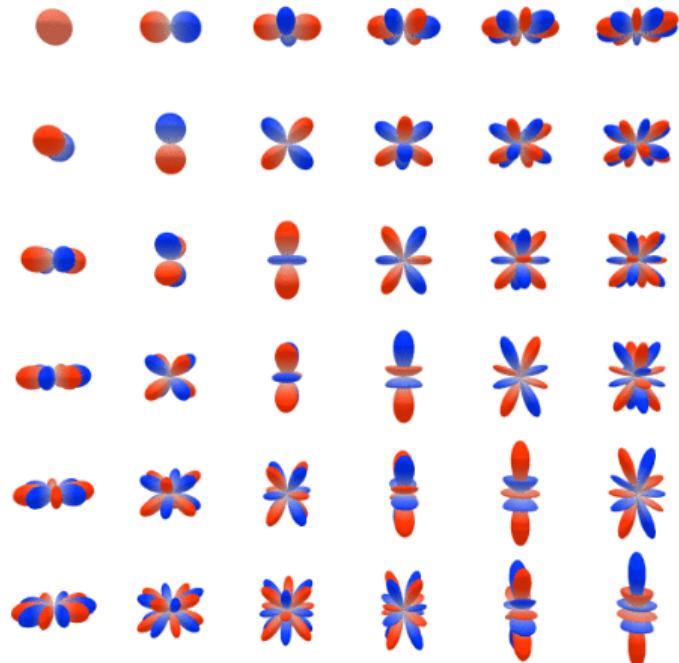
- Equivariance:

$$f(Tx) = Tf(x)$$



Equivariance vs Invariance

Use spherical harmonics in the message passing to ensure that **only certain symmetries are allowed**.



$$m_m^{(l)} = \sum_{m_1, m_2} c_{l_1 m_1, l_2 m_2}^{lm} h_{m_1}^{(l_1)} r_{m_2}^{(l_2)}$$

$$r_{ij} = Y_m^{(l)}(r_{ij}) f(r_{ij})$$

GRAPH NETWORKS AND THE MATBENCH DATASET

Graph neural networks are widely used for property predictions in chemistry but excel on larger datasets

	Target name (unit)												
	Yield Strength (GPa)	E Exfol. (meV/atom)	PhDOS Peak (1/cm)	(no unit)	Band Gap (eV)	Metallicity	GFA	log K (log(GPa))	log G (log(GPa))	E form. (eV/atom)	Band Gap (eV)	Metallicity	E form. (eV/atom)
Algorithm	97.5	39.9	56.2	0.315	0.294	0.934	0.88	0.0647	0.0874	0.2	0.282	0.909	0.173
Automatminer	97.5	39.9	56.2	0.315	0.294	0.934	0.88	0.0647	0.0874	0.2	0.282	0.909	0.173
RF	104	49.9	68	0.421	0.356	0.929	0.875	0.081	0.104	0.235	0.345	0.9	0.116
CGCNN		49.2	57.8	0.599				0.0712	0.0895	0.0452	0.228	0.954	0.0332
MEGNet		55.9	39.4	0.543				0.0751	0.0939	0.0417	0.222	0.977	0.0389
Best Literature*		37.3		0.5		0.97	0.8						

USES OF GRAPH NETWORKS

GNNs take up **most of the top spots** on the current leader board

Many **high-performance MLIPs** use graphs (MACE, nequip, allegro)

Task name	Samples	Algorithm	Verified MAE (unit) or ROCAUC	Notes
matbench_steels	312	MODNet (v0.1.12)	87.7627 (MPa)	
matbench_jdft2d	636	MODNet (v0.1.12)	33.1918 (meV/atom)	
matbench_phonons	1,265	MegNet (kgcnn v2.1.0)	28.7606 (cm^-1)	structure required
matbench_expt_gap	4,604	MODNet (v0.1.12)	0.3327 (eV)	
matbench_dielectric	4,764	MODNet (v0.1.12)	0.2711 (unitless)	
matbench_expt_is_metal	4,921	AMMExpress v2020	0.9209	
matbench_glass	5,680	MODNet (v0.1.12)	0.9603	
matbench_log_gvrh	10,987	coNGN	0.0670 (log10(GPa))	structure required
matbench_log_kvrh	10,987	coNGN	0.0491 (log10(GPa))	structure required
matbench_perovskites	18,928	coGN	0.0269 (eV/unit cell)	structure required
matbench_mp_gap	106,113	coGN	0.1559 (eV)	structure required
matbench_mp_is_metal	106,113	CGCNN v2019	0.9520	structure required
matbench_mp_e_form	132,752	coGN	0.0170 (eV/atom)	structure required

CONCEPT CHECKLIST

- Many datasets can be represented as graphs.
- GNNs work by i) building a graph and ii) propagating information between neighbours using NNs
- GNNs are scalable and can generalise well
- There are many possibilities for designing GNNs



THANK YOU

mdi-group.github.com