A gentle introduction to graph neural networks

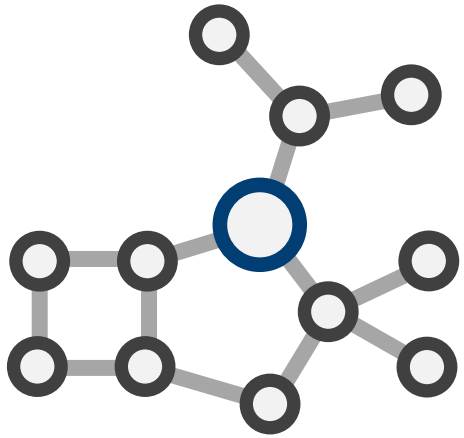
Alex Ganose

Department of Chemistry

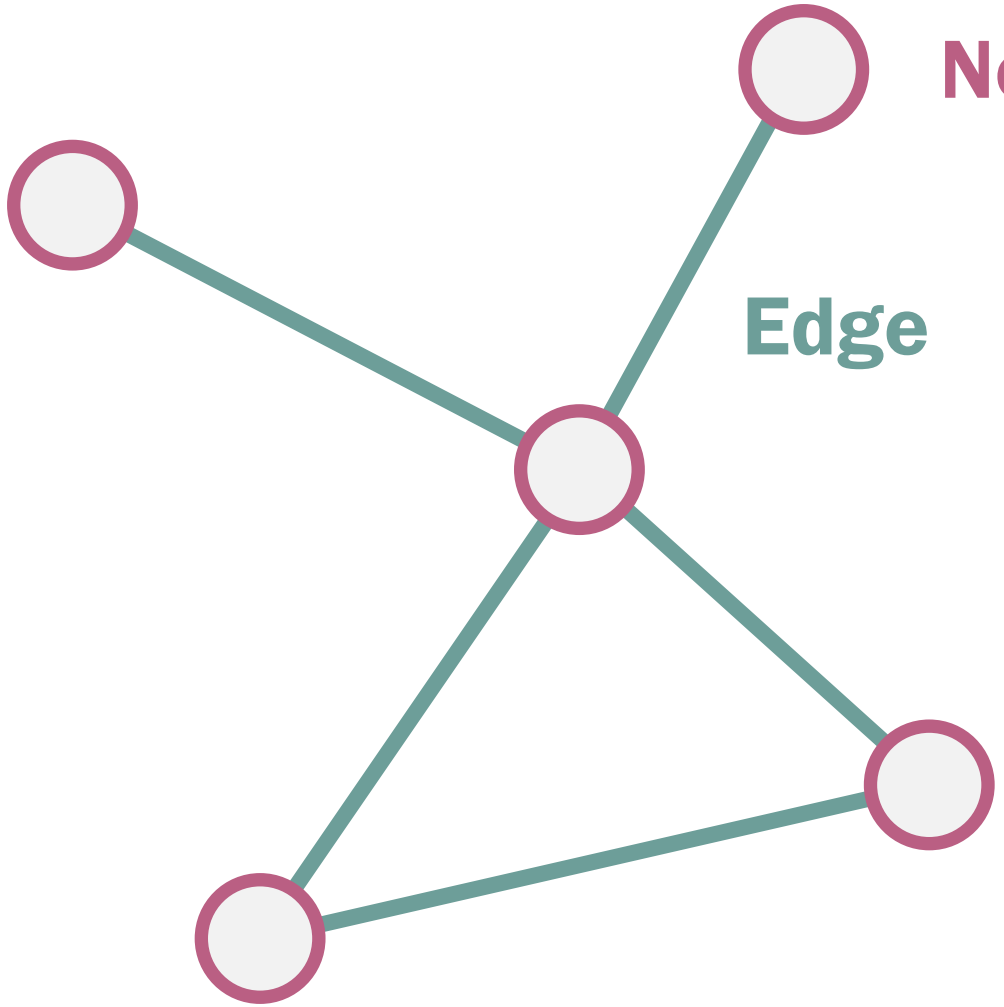
Imperial College London

a.ganose@imperial.ac.uk

website: virtualatoms.org



What is a graph?

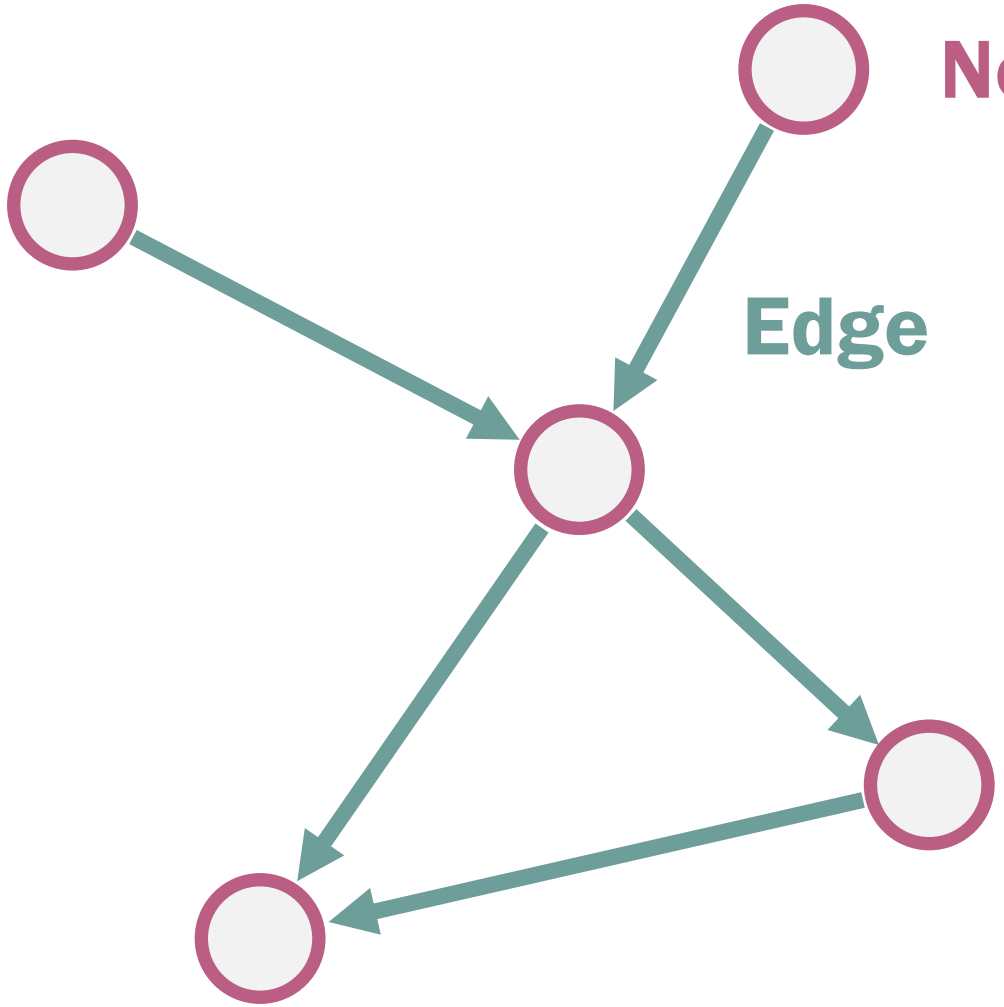


Node / vertex

Edge

Graphs encode relations between entities

What is a graph?

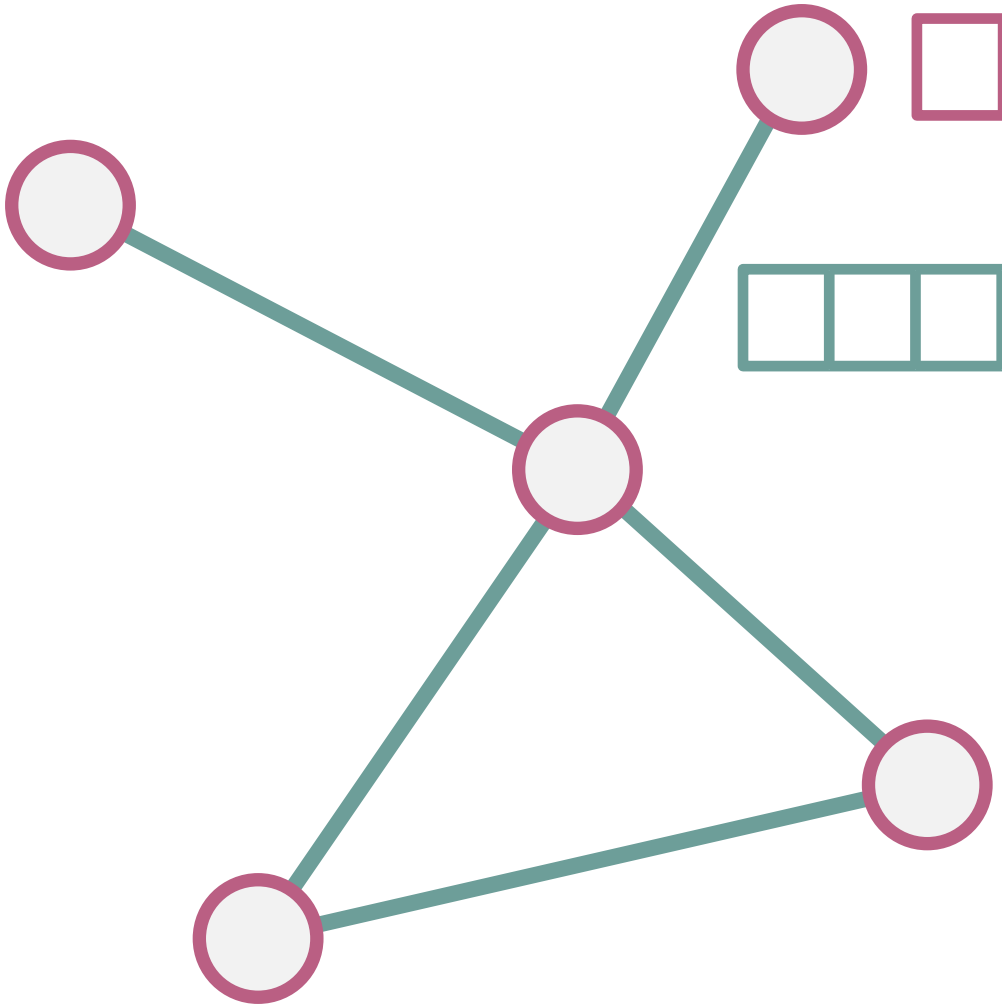


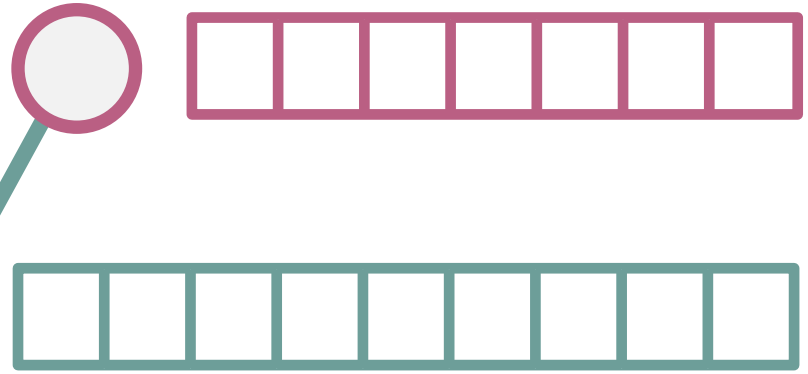
Node / vertex

Edge

Edges can be directed

What is a graph?



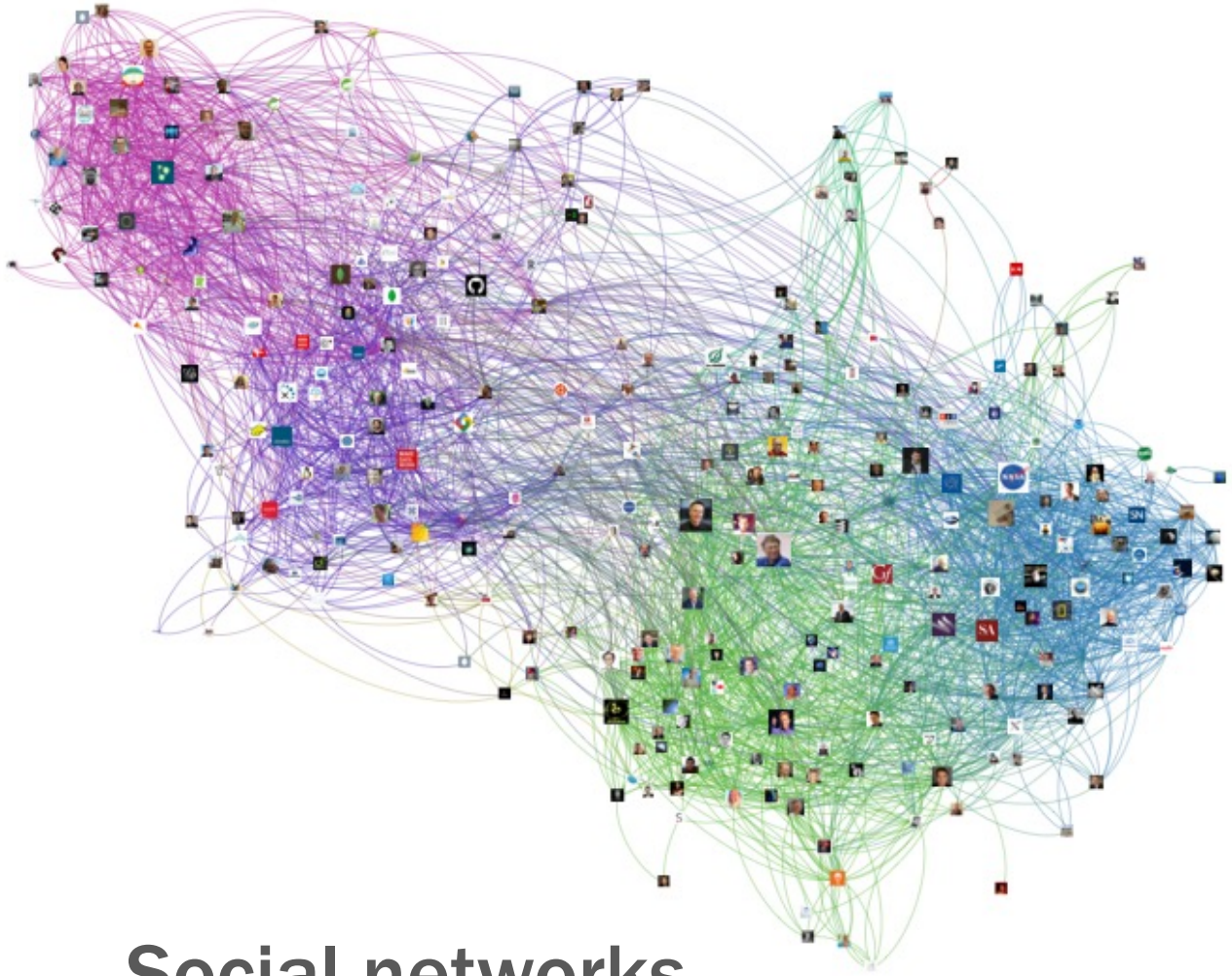


Node embedding

Edge embedding

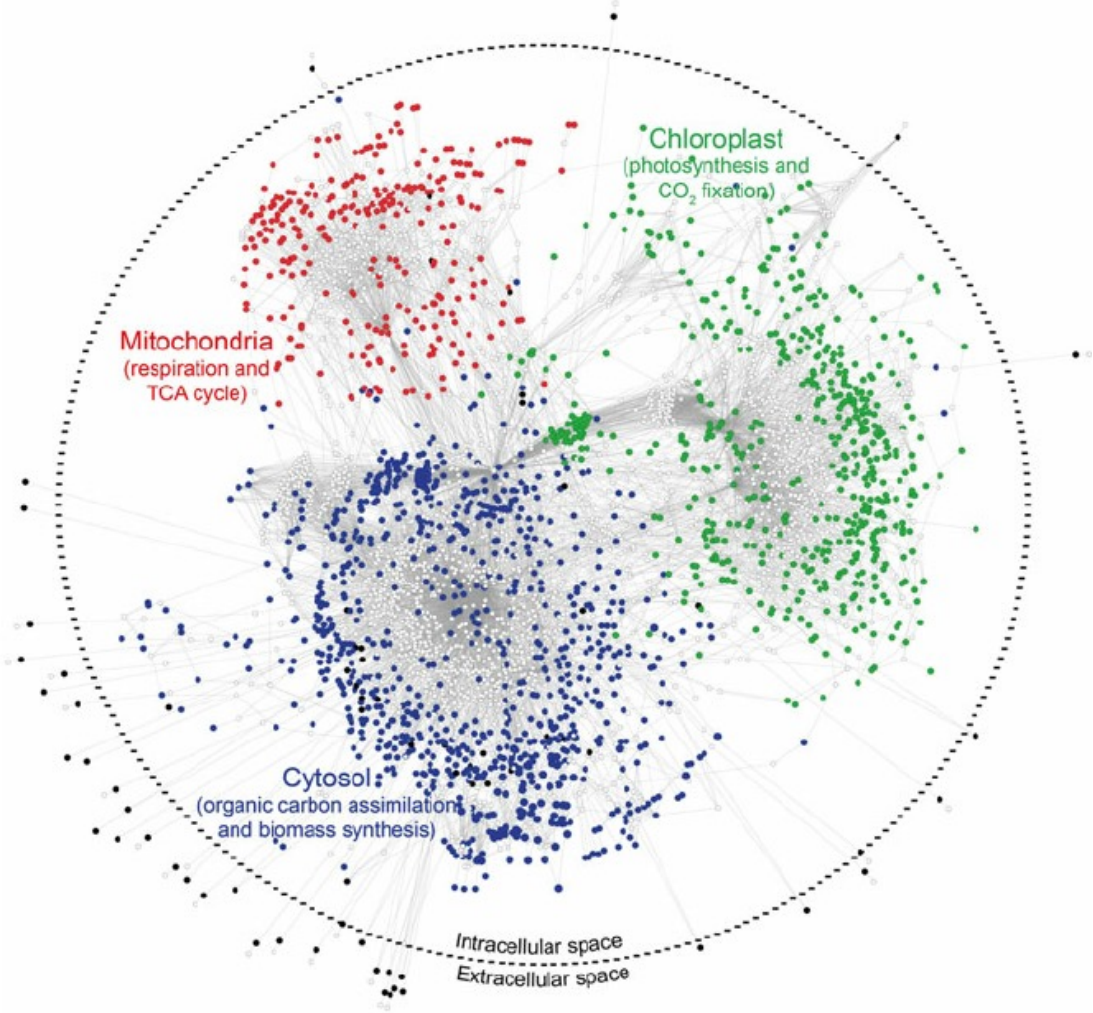
Information is stored in each piece

Where do we find graphs



Social networks

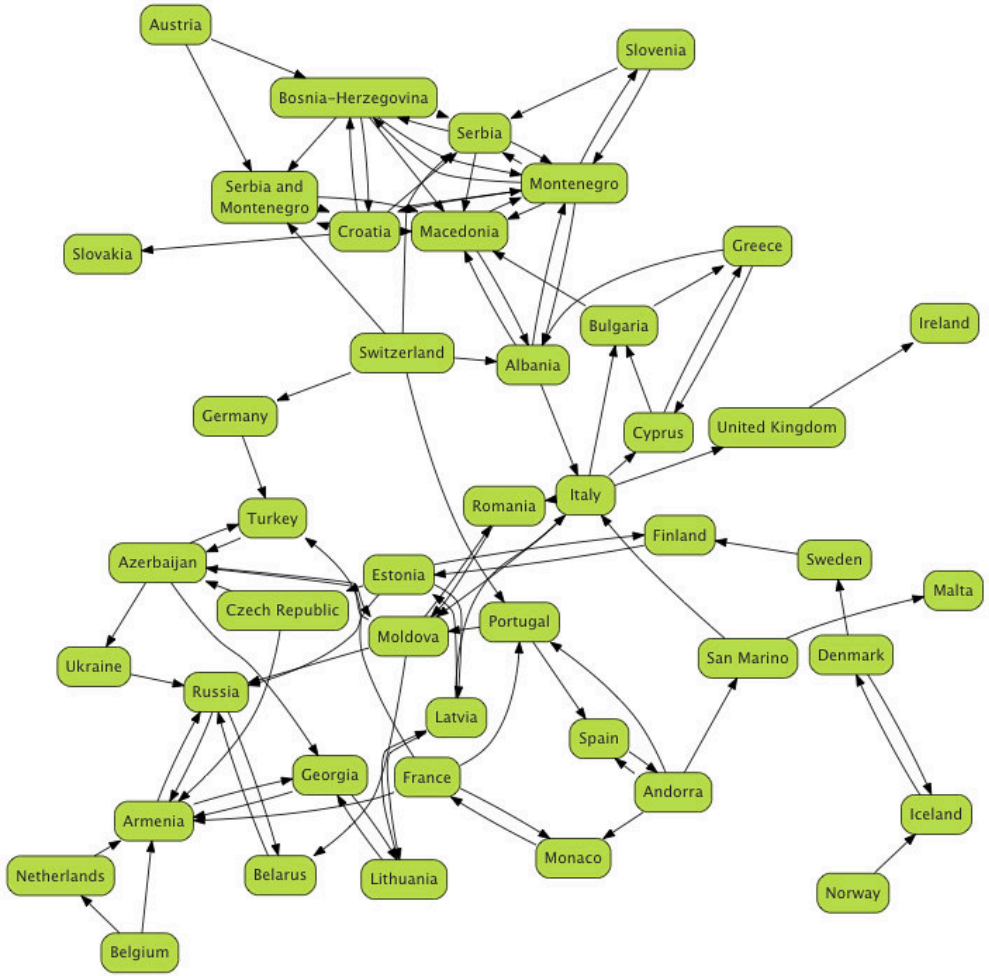
> 1B nodes > 10B edges



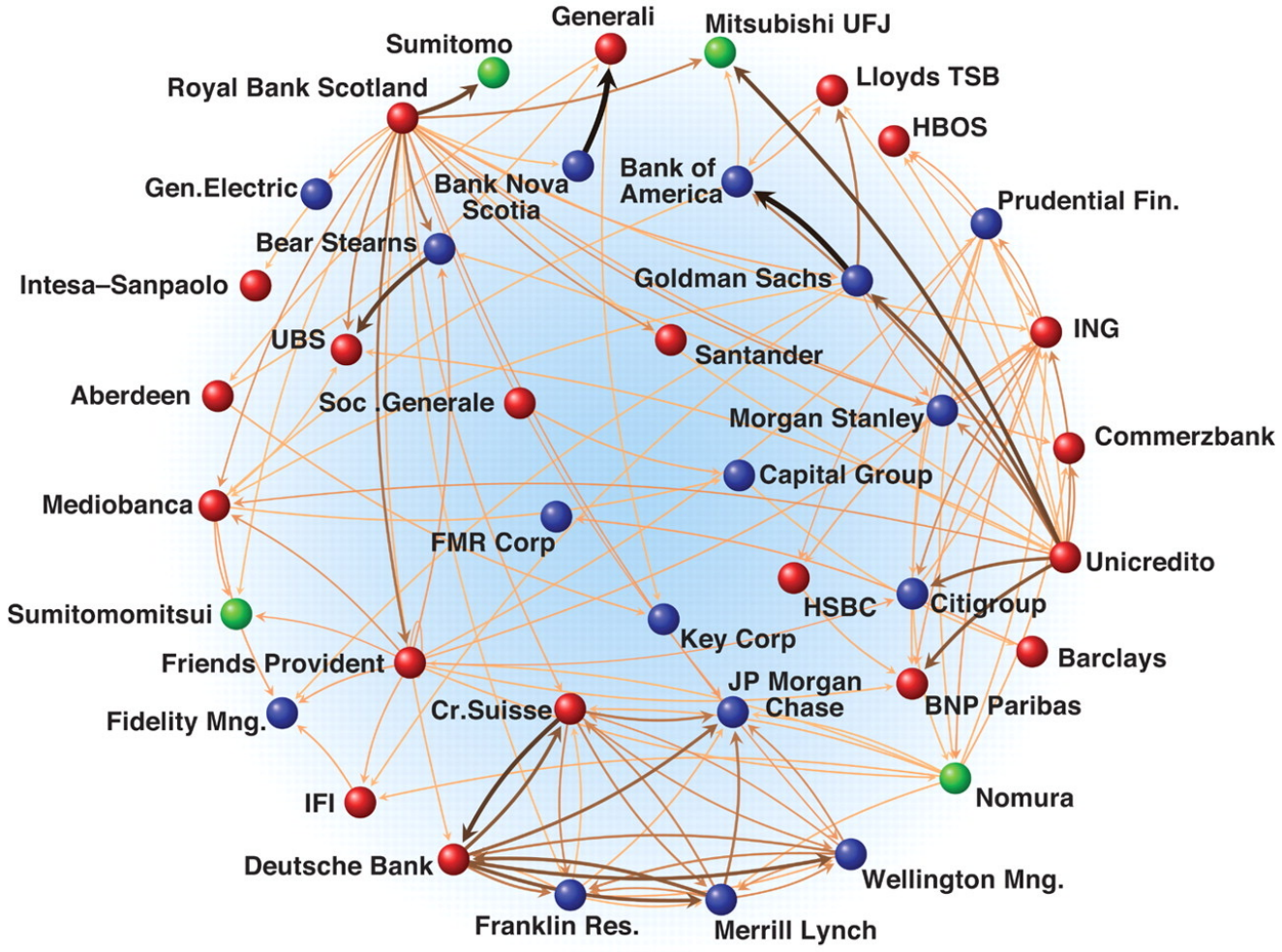
Biological systems

Clamydomonas reinhardtii

Where do we find graphs



Eurovision



Economics

An image is a graph with regular structure

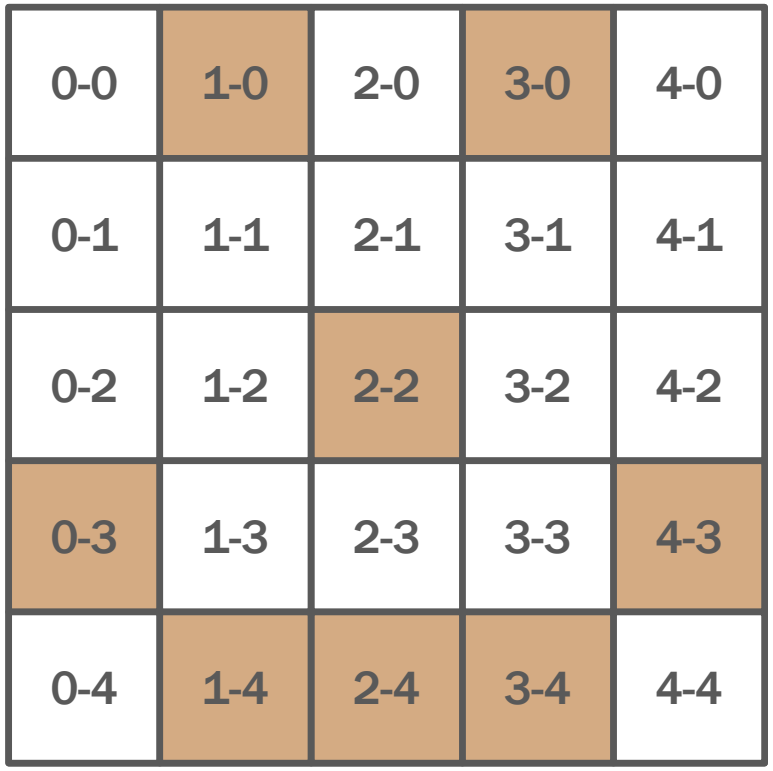
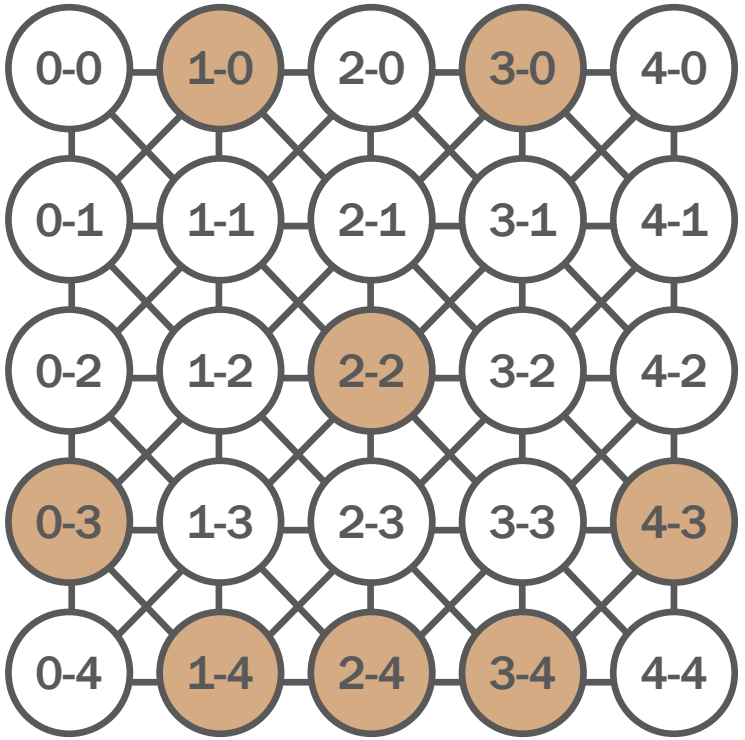
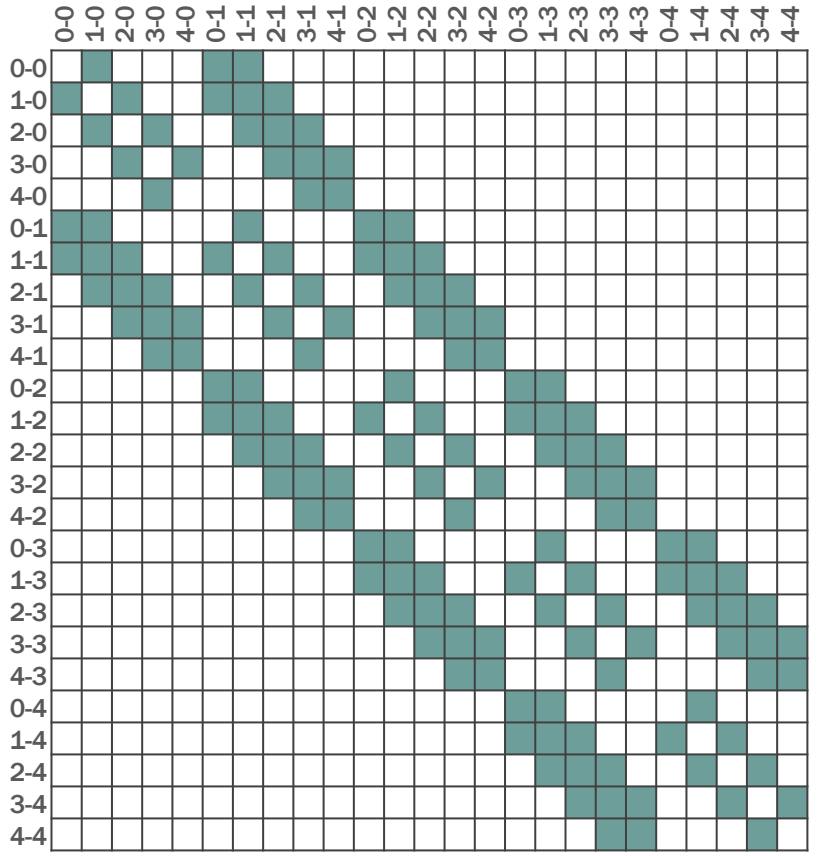


Image pixels

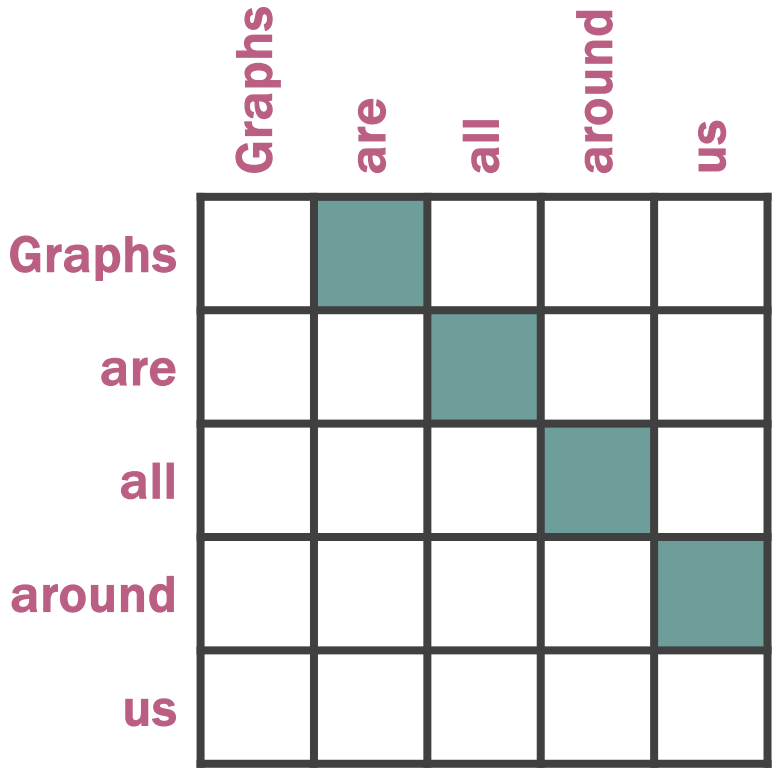


Graph

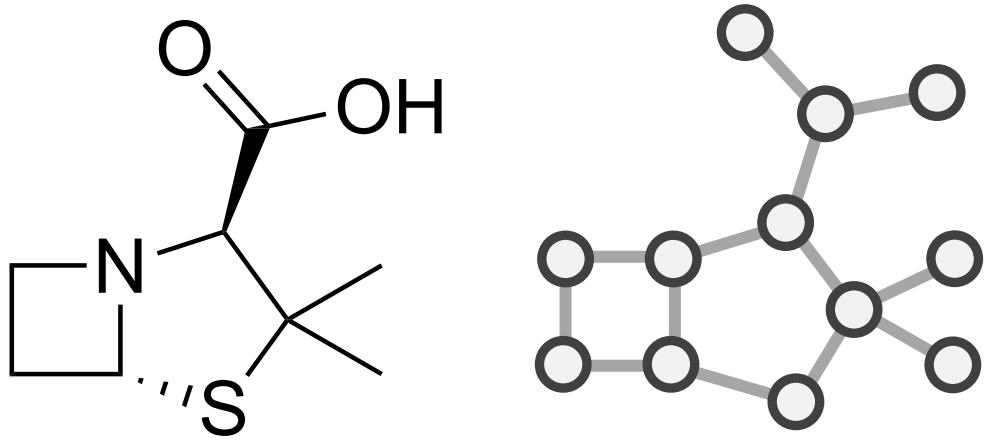


Adjacency matrix

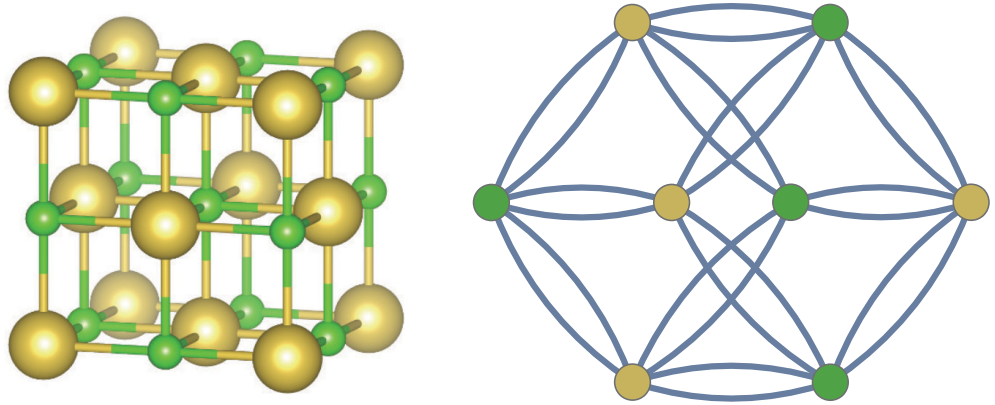
A sentence can be viewed as a directed graph



Graphs are a natural representation in chemistry



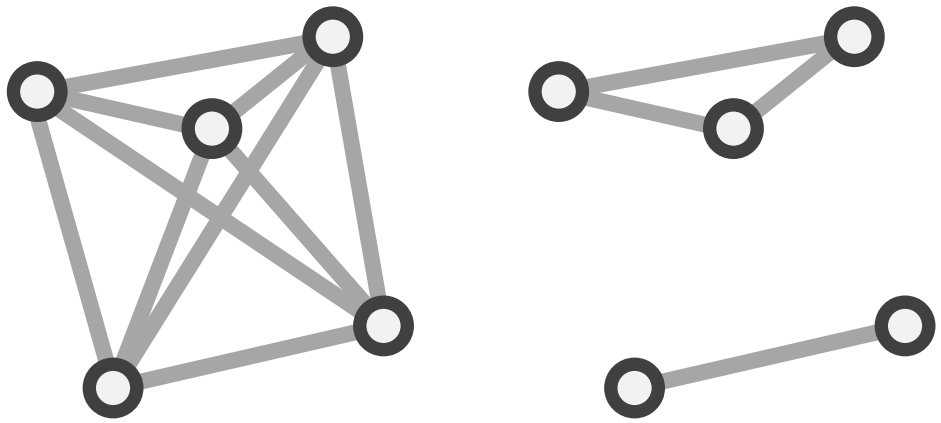
Molecules



Crystals

All graphs are not alike

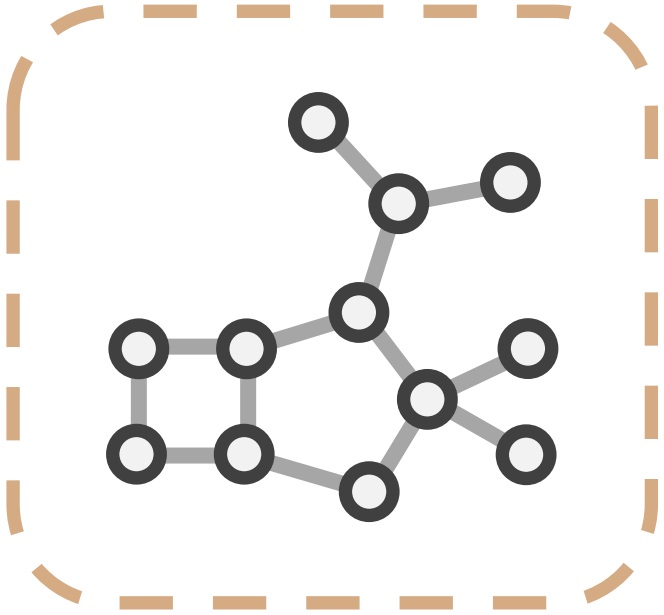
The size and connectivity of graphs can vary enormously



Fully connected Sparse

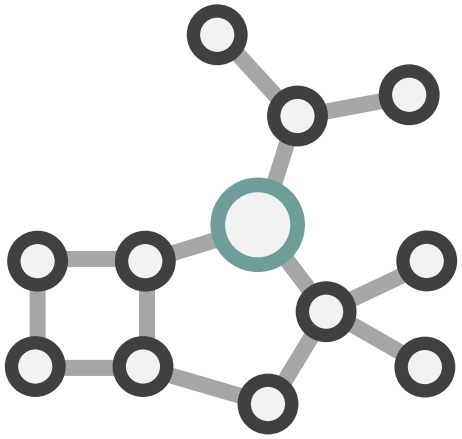
Dataset	Graphs	Nodes	Edges
Fully con.	1	5	20
Sparse	2	<4	<3
Wikipedia	1	12M	378M
qm9	134k	<10	<26
Cora	1	23k	91k

The types of problems tackled with graphs



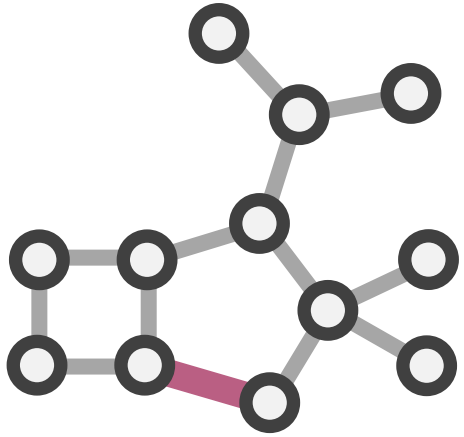
Graph level

e.g. total energy of a molecule



Node level

e.g. oxidation state of an atom

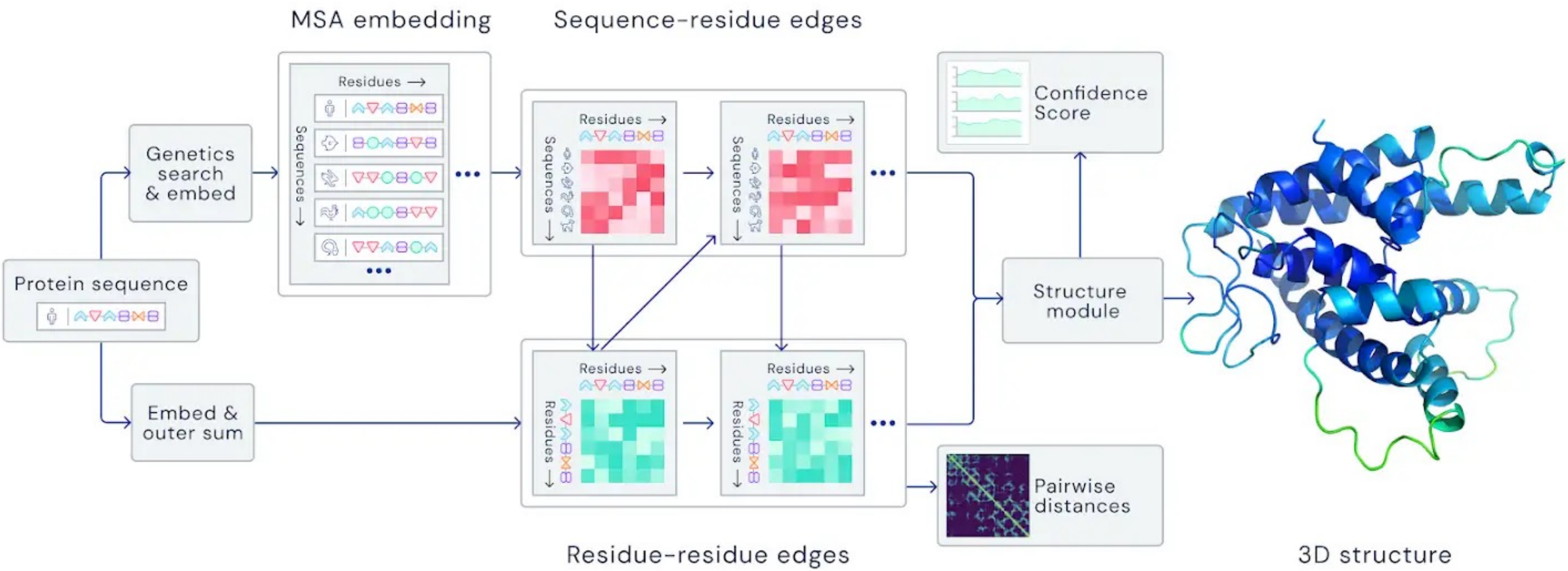


Edge level

e.g. strength of a bond

Graph networks enabled Alpha Fold (node level)

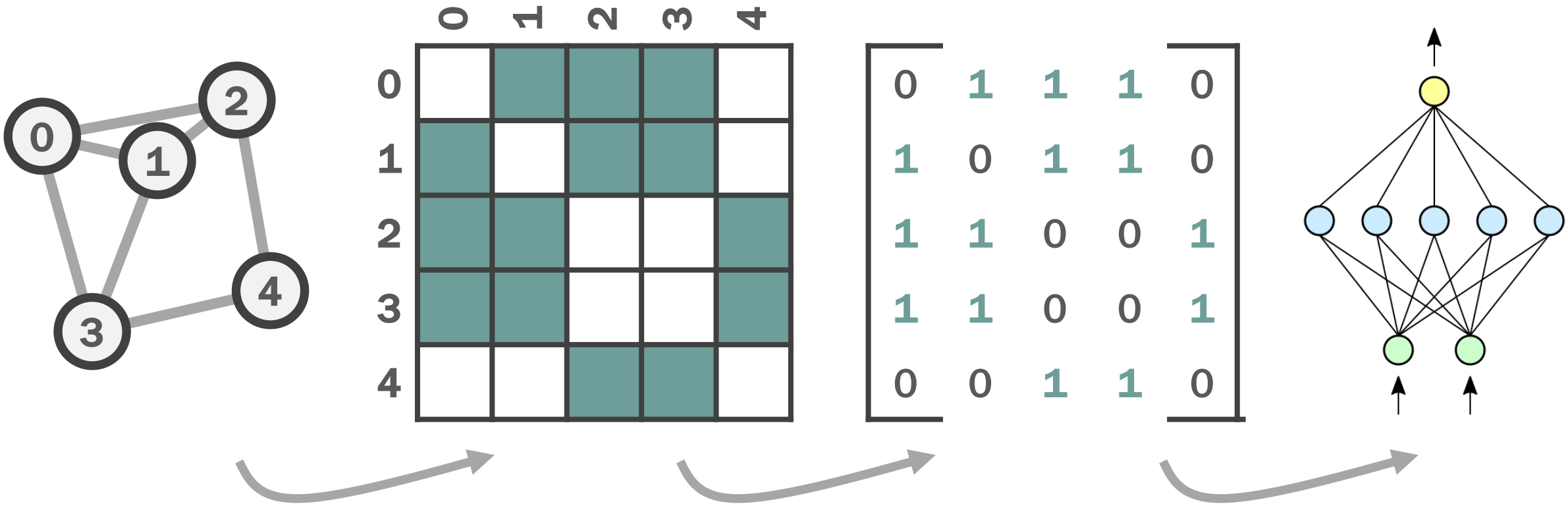
Protein as a graph with amino acids (nodes) linked by edges



Used to calculate interactions between parts of the protein

Deep learning with graphs

Include adjacency matrix as features in a standard neural network

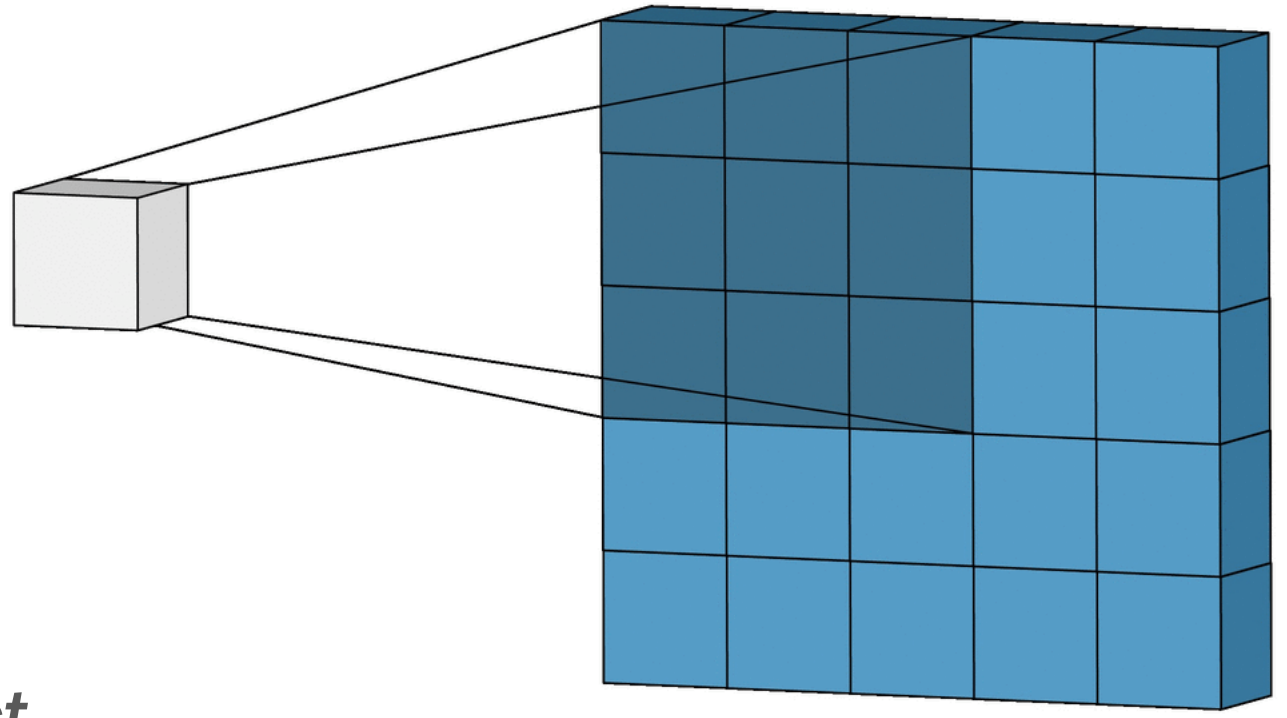


Issues: fixed size and sensitive to the order of nodes

Deep learning with 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

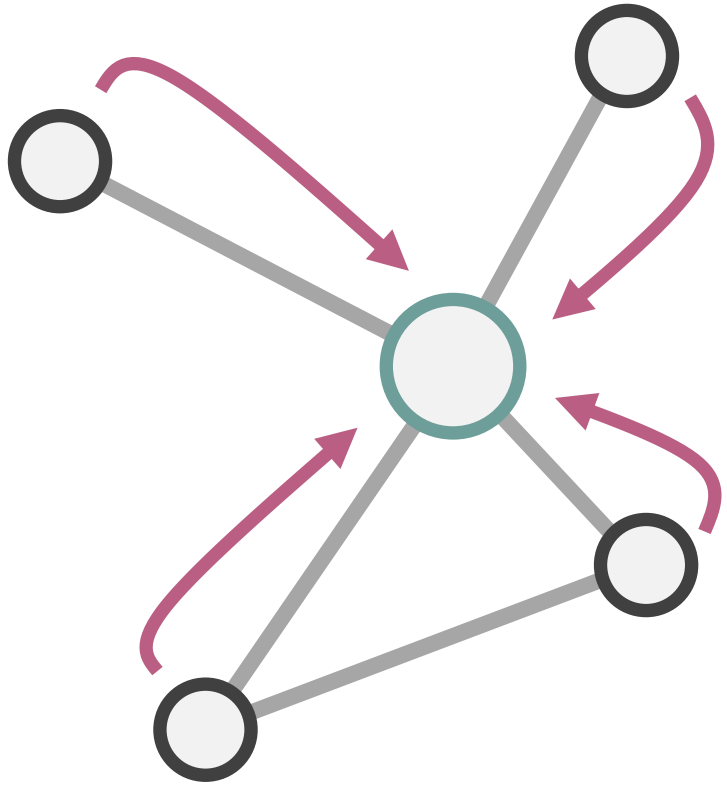


Convolution filter

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

Convolutions on graphs

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

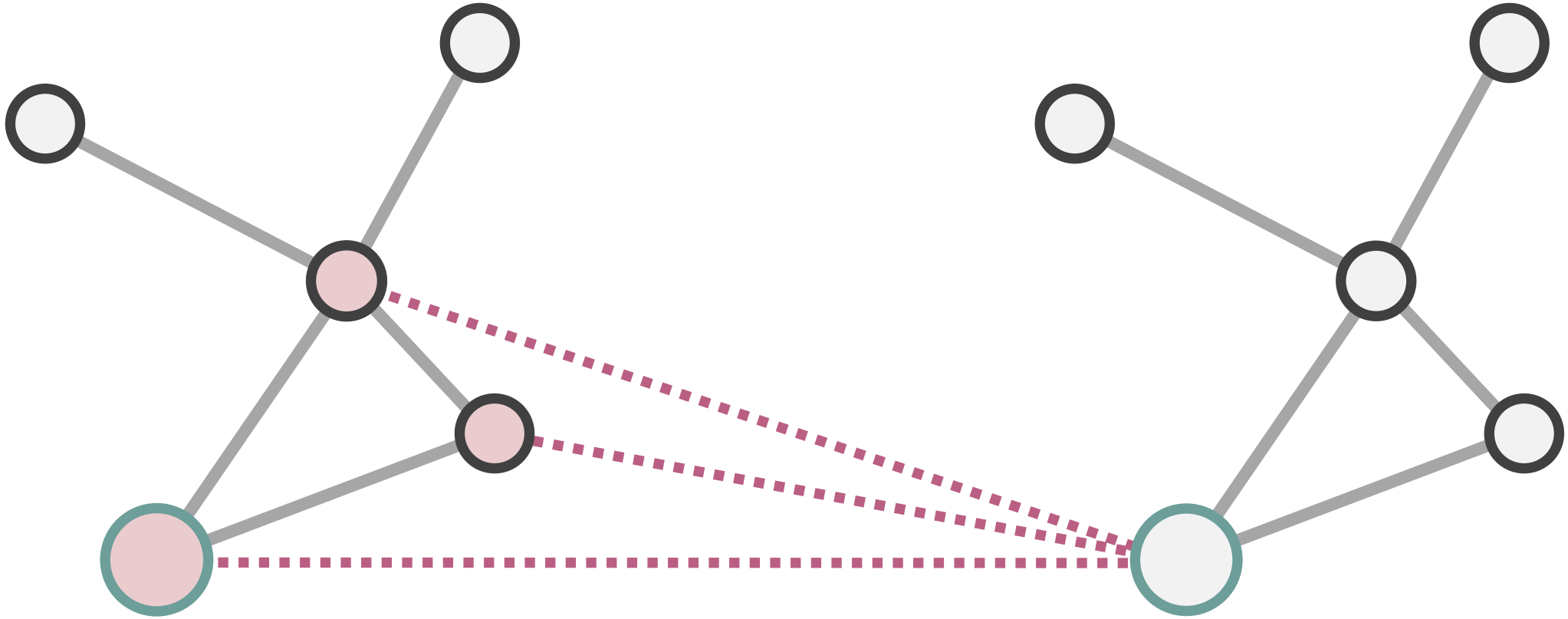


Convolution from neighbours

to central node

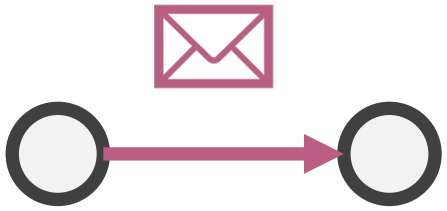
Convolutions on graphs

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



Where do neural networks come in?

Neural networks are used to decide:

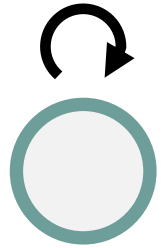


Message

What get passed from one node to another

Pooling / Aggregation

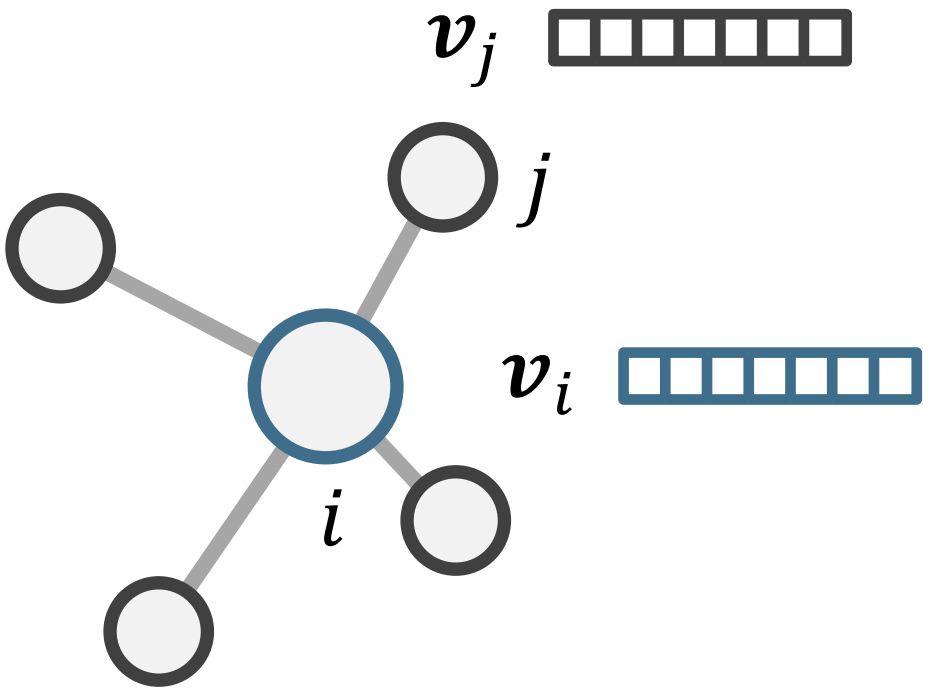
How messages from all neighbours are combined



Update

How the node is updated given the pooled message

Components of a convolutional graph network



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

Message function

Pooling function

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

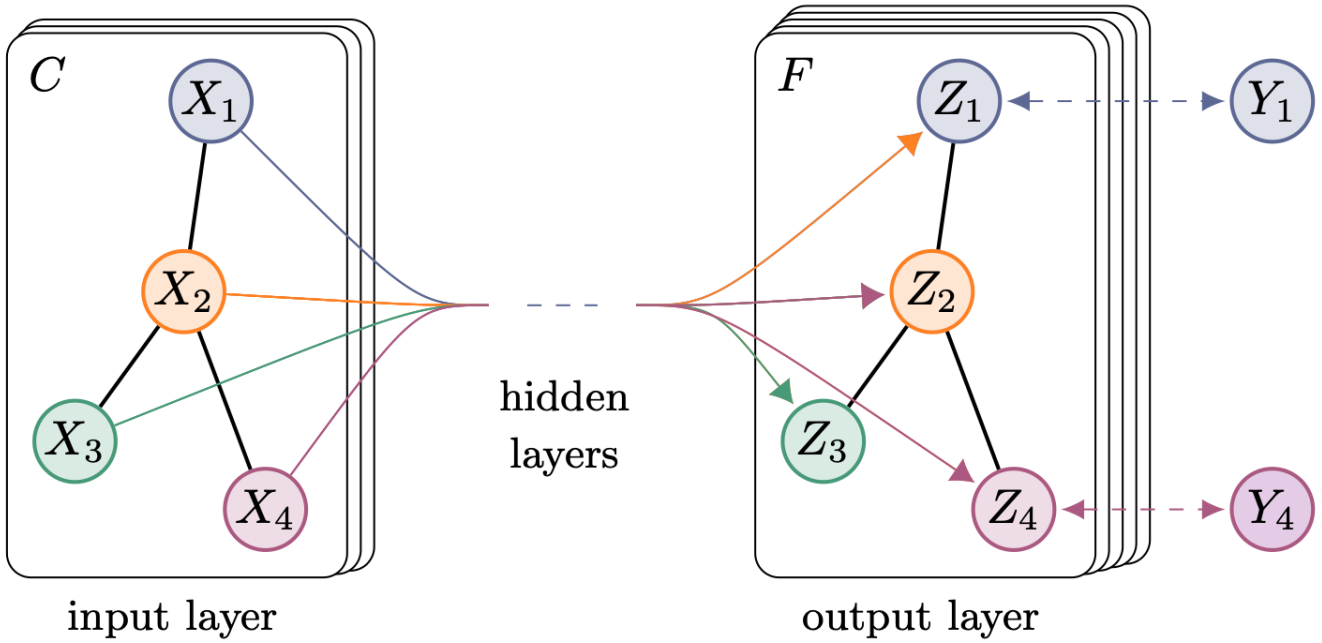
Update function

Convolutional graph networks introduced in 2017

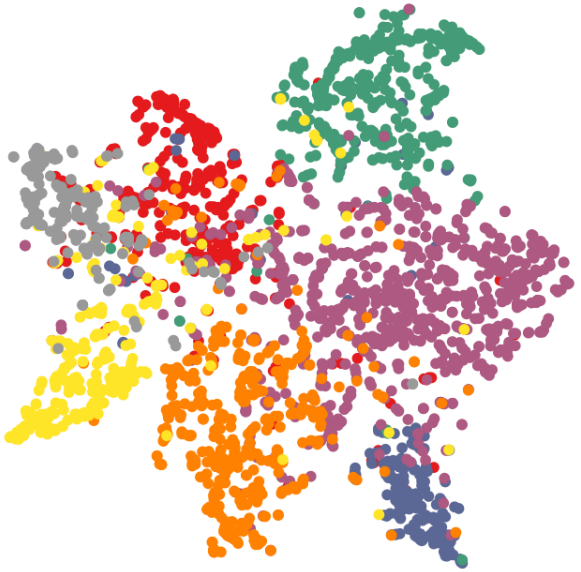
SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

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



(a) Graph Convolutional Network



(b) Hidden layer activations

Implementation of neural network functions

Message function: $\boldsymbol{v}_j$ *(no processing)*

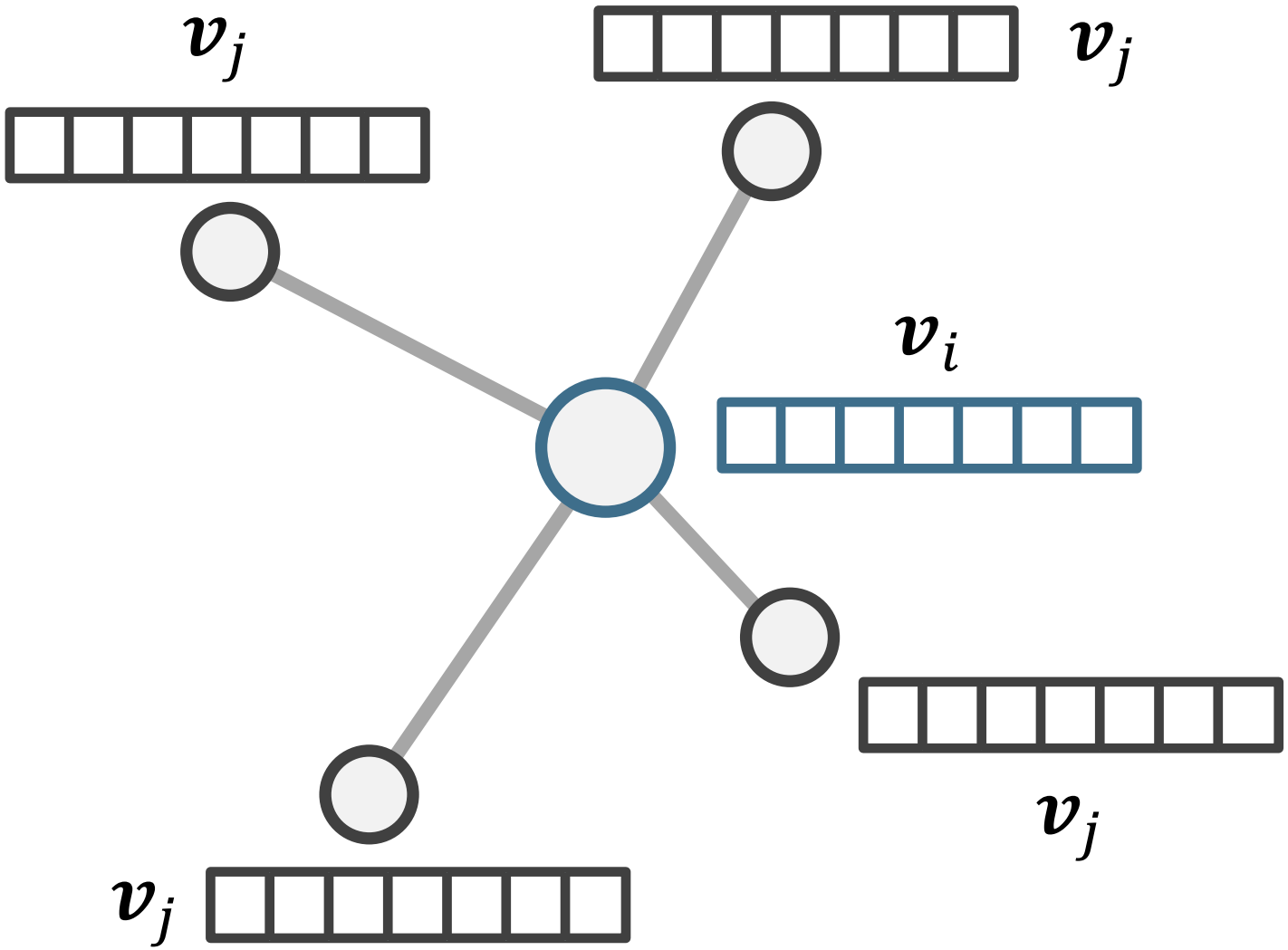
Pooling function: $\boldsymbol{m}_i = \sum_{j \in \mathcal{N}(i)} \frac{\boldsymbol{v}_j}{|\mathcal{N}(i)|}$ *(normalised sum)*

num neighbours

Update function: $\boldsymbol{v}'_i = \sigma(\mathbf{W}\boldsymbol{m}_i + \mathbf{B}\boldsymbol{v}_i)$ *(MLP)*

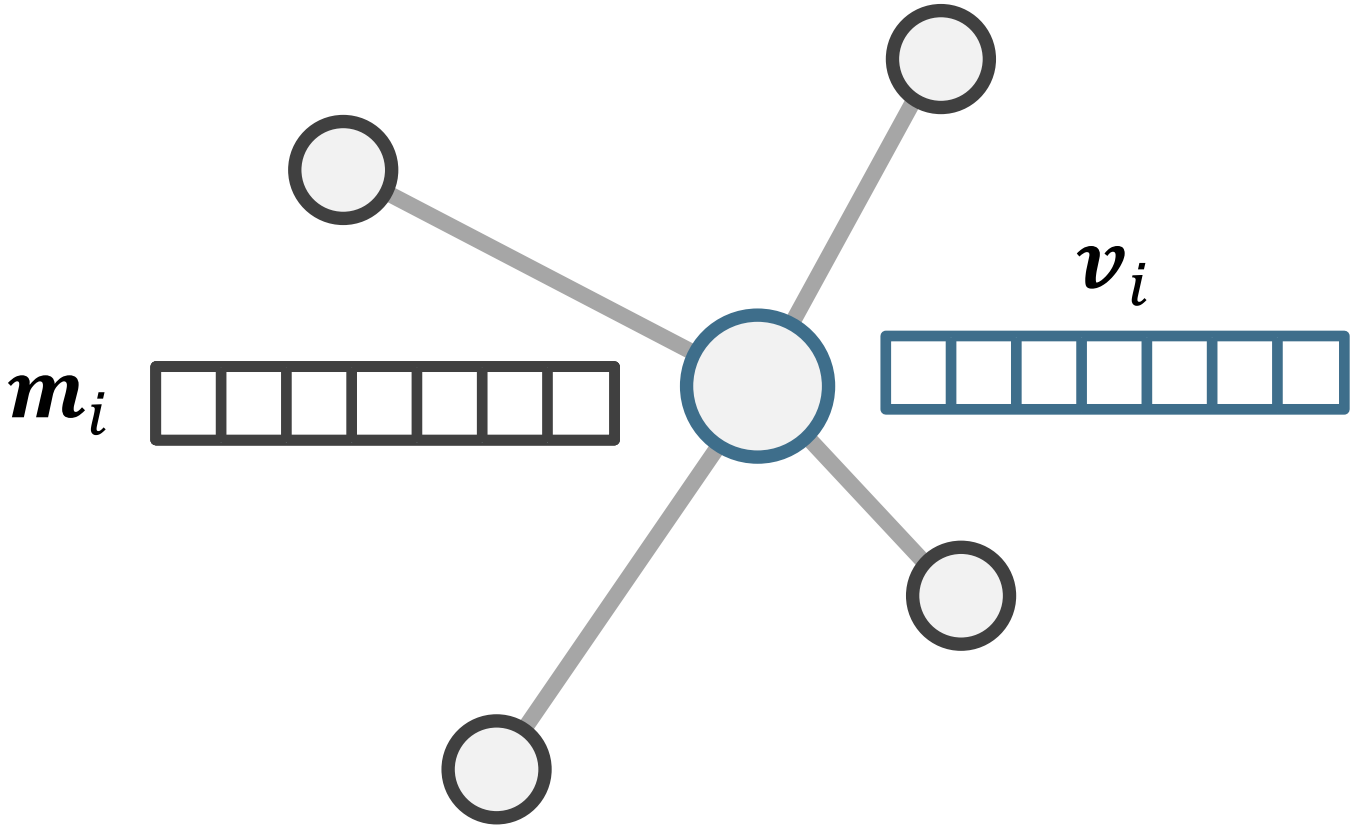
non-linearity *weights*

Visual depiction of a graph convolution



1. Prepare messages

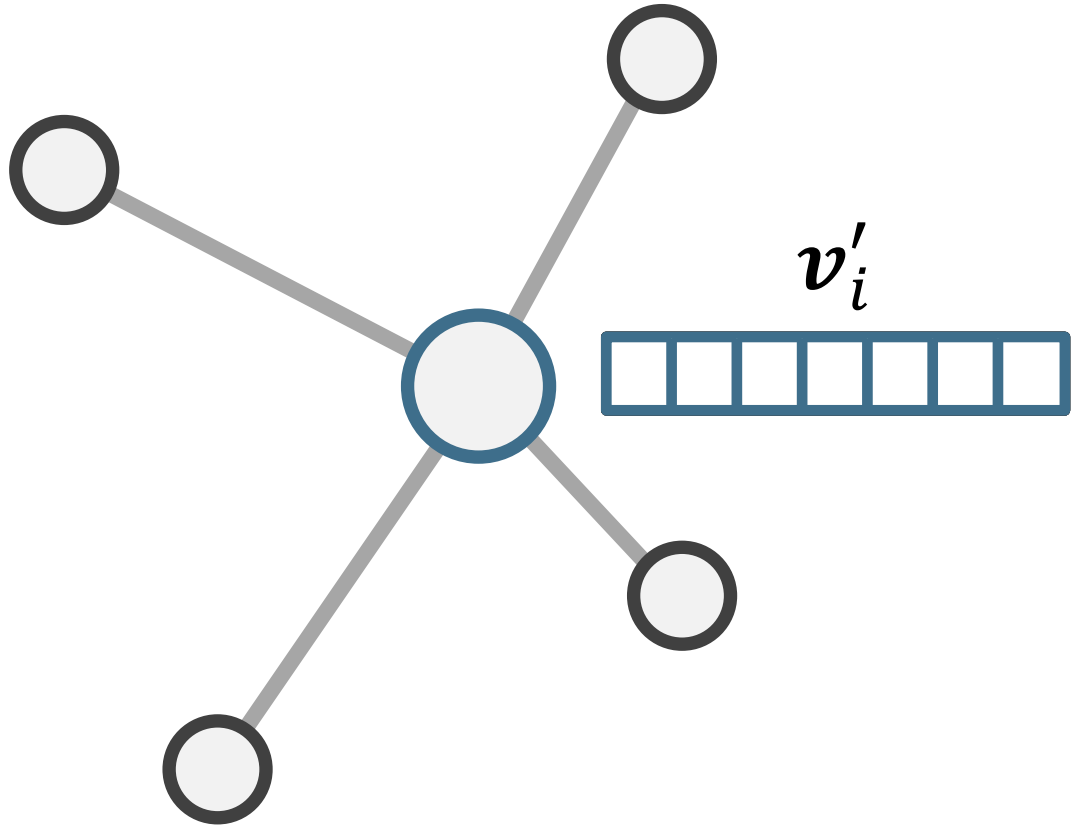
Visual depiction of a graph convolution



1. Prepare messages

2. Pool messages

Visual depiction of a graph convolution



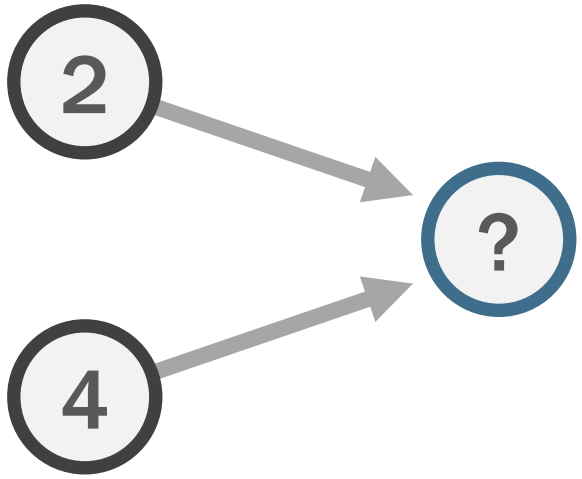
1. Prepare messages

2. Pool messages

3. Update embedding

Requirements 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

All take a variable number of inputs and provide an output that is the same, no matter the ordering

Training convolutional graph neural networks

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

Feed the final node embeddings to a loss function

Run an optimiser to train the weight parameters

$\mathbf{W}$ and $\mathbf{B}$ are shared across all nodes

Inductive capabilities and efficiency

Each node has its own network due to its connectivity

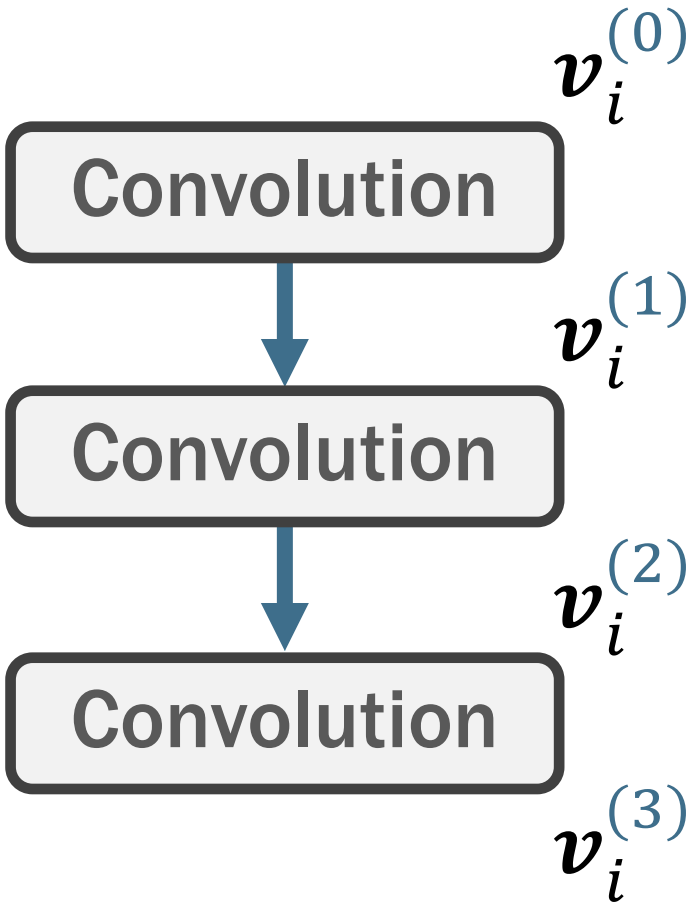
Message, pool, and update functions are shared for all nodes

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 multiple convolutional layers

Only looked at a single convolution – can we stack multiple layers?

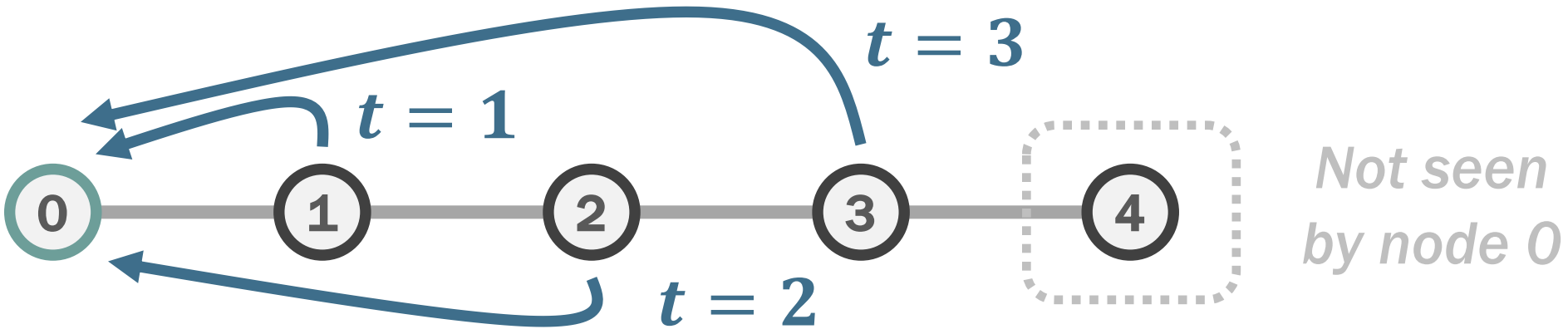


$$\boldsymbol{v}_i^{(t+1)} = \sigma\left(\mathbf{W}^{(t)} \sum_{j \in \mathcal{N}(i)} \frac{\boldsymbol{v}_j^{(t)}}{|\mathcal{N}(i)|} + \mathbf{B}^{(t)} \boldsymbol{v}_i^{(t)}\right)$$

Weights are unique for each layer

Why multiple convolutions?

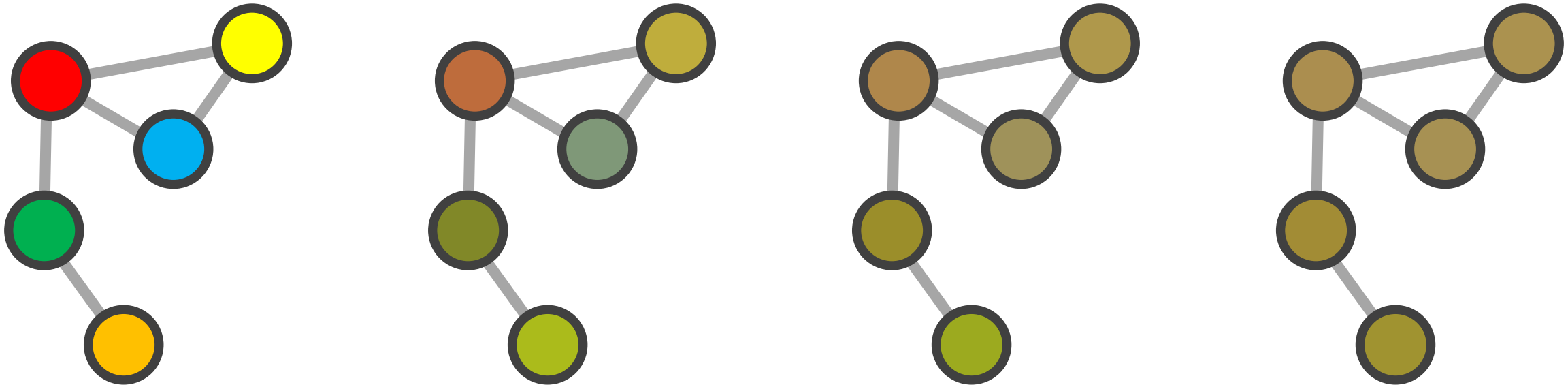
Graph are inherently local – Nodes can only see other nodes t convolutions away



Multiple convolutions increases the "receptive field" of the nodes

The over smoothing problem

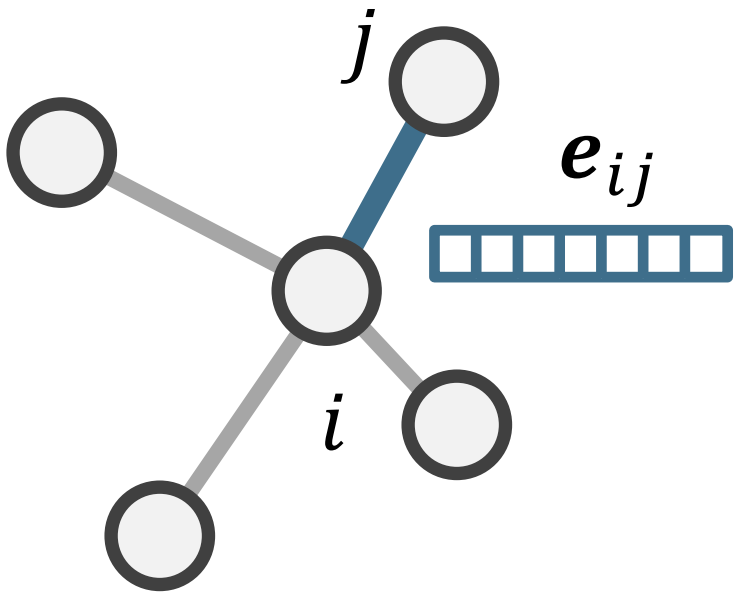
However, too many convolutions causes over smoothing — all node embeddings converge to the same value



$t = 0$ $t = 1$ $t = 2$ $t = 3$

What about edge embeddings

Only considered node updates but graphs have edges too — can we learn something about edges from nodes?



Edge embedding

$$\boldsymbol{m}_i = \bigoplus_{j \in \mathcal{N}(i)} M_t(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{e}_{ij})$$

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

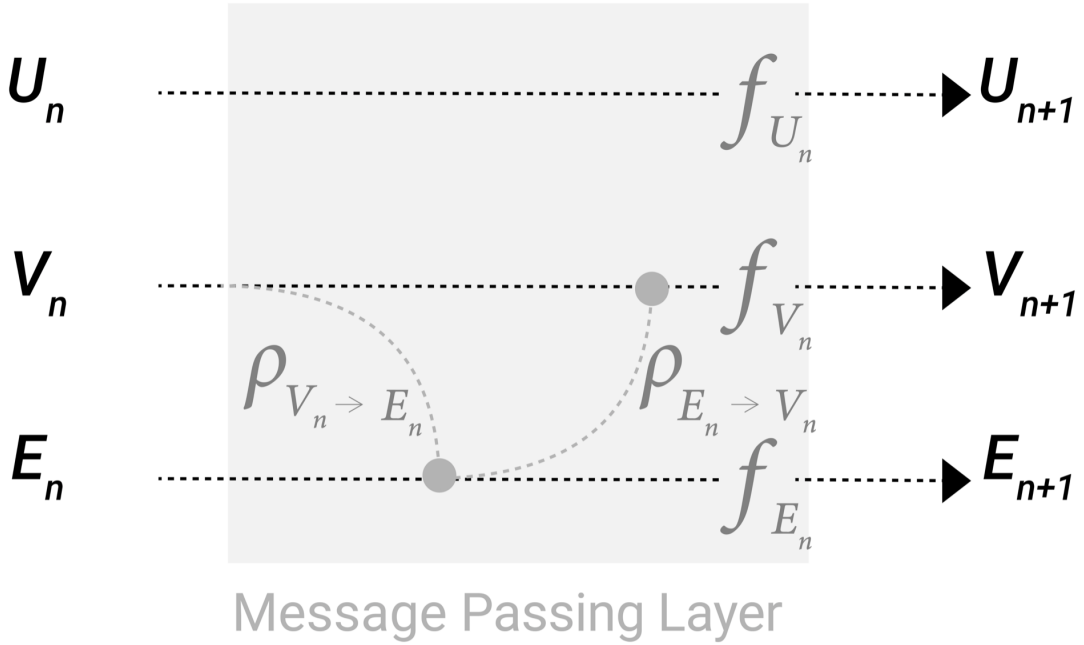
Update function stays the same

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

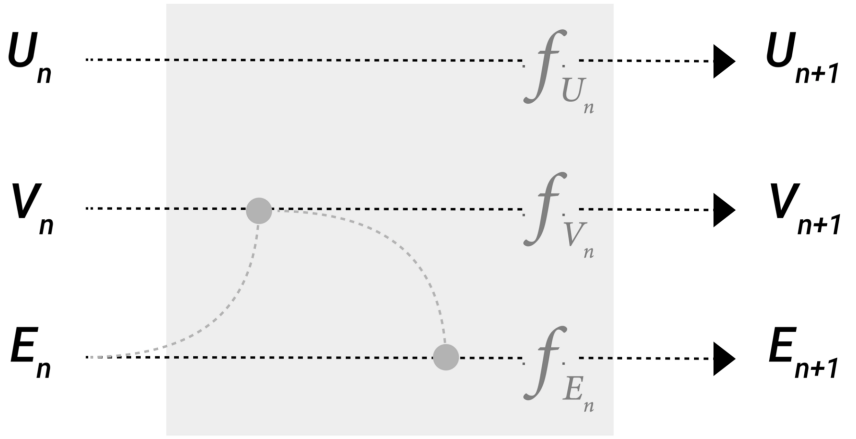


Message passing networks – significant flexibility

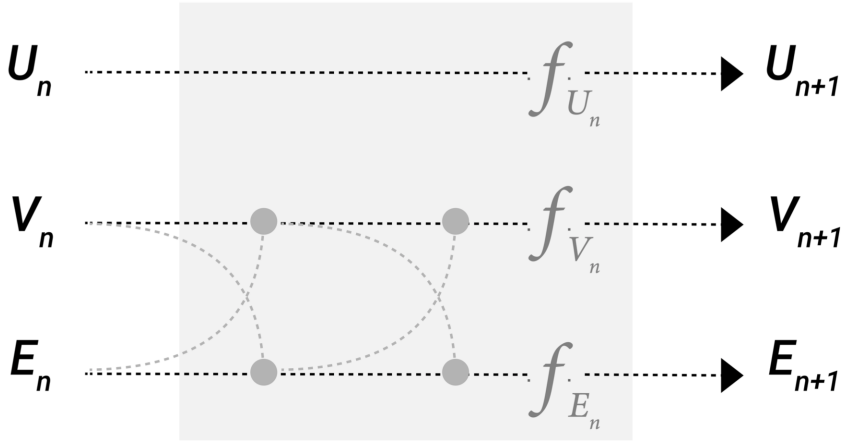
Can update nodes before edges or vice versa

Or have a weave design to pass messages back and forth

All flexible design choices in message passing networks



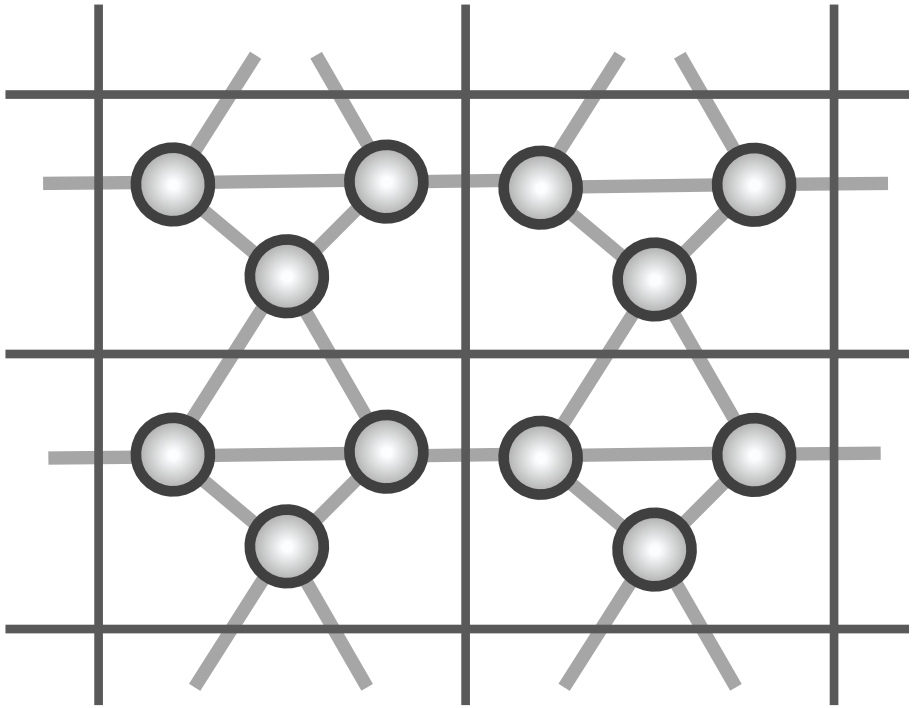
Node Then Edge Learning



Weave Layer

Convolutional graph networks for crystals

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

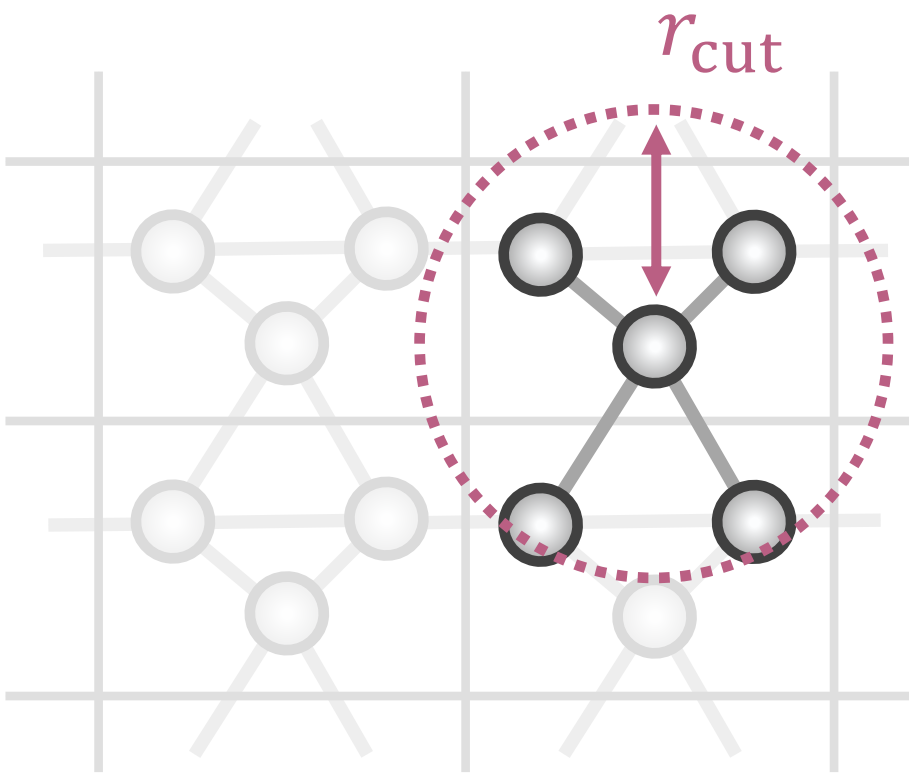


Networks should be permutation and translation invariant

Properties depend on atom types and coordinates not just connectivity

Constructing the graph from a crystal structure

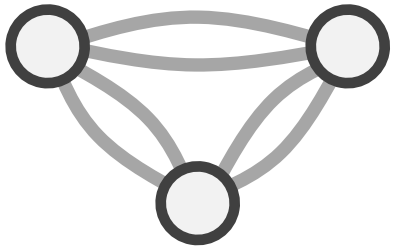
Include all atoms within a certain cut-off as neighbours



Must consider periodic boundaries

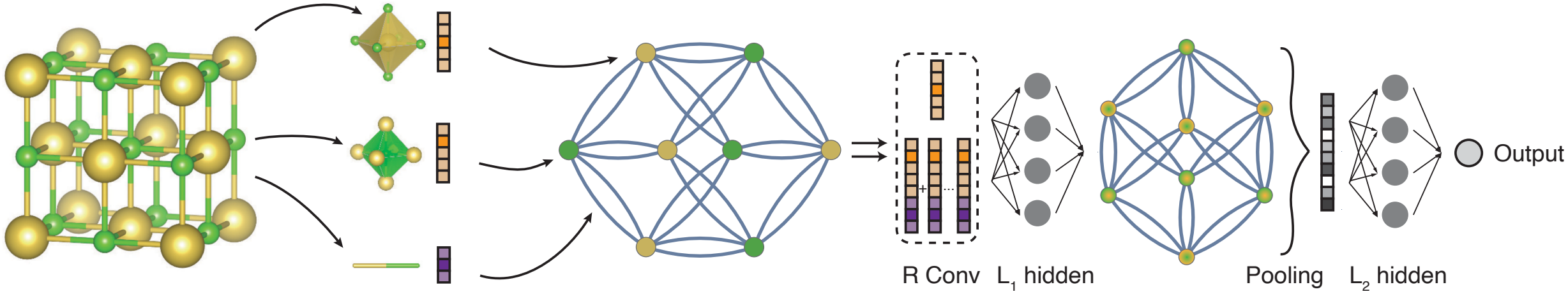
Perform the procedure for each atom in the unit cell

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



Xie and Grossman *Phys. Rev. Lett.* **120**, 145301 (2018)

Implementation of CGCNN

Message function:

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

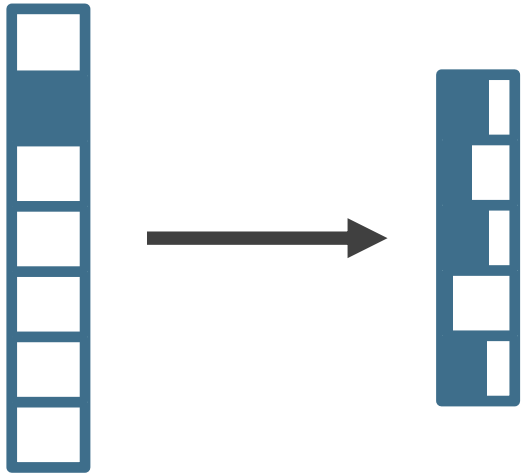
Update function:

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

sigmoid *"gate"* *softplus*

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

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

num atoms

MLP readout

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

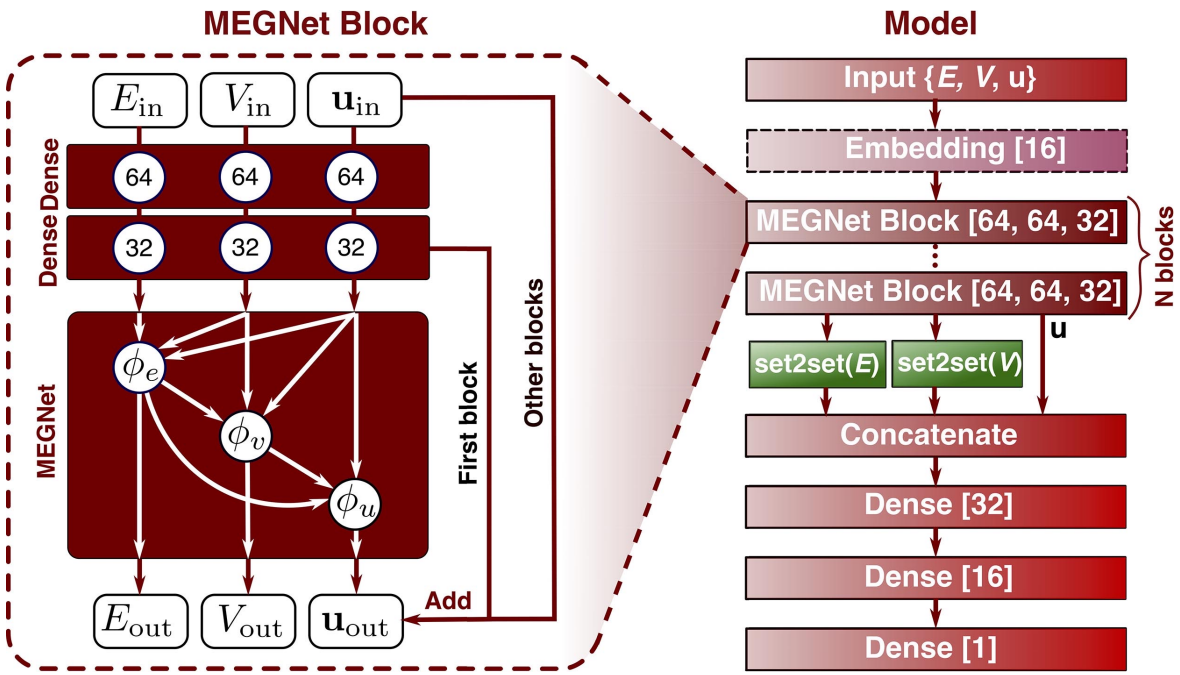
CGCNN performance

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	···

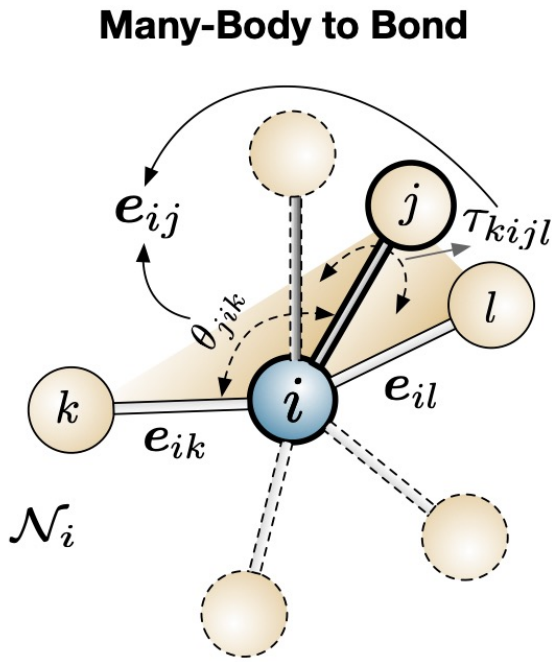
Advanced message passing networks

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



MEGNet

Skip connections and set2set pooling

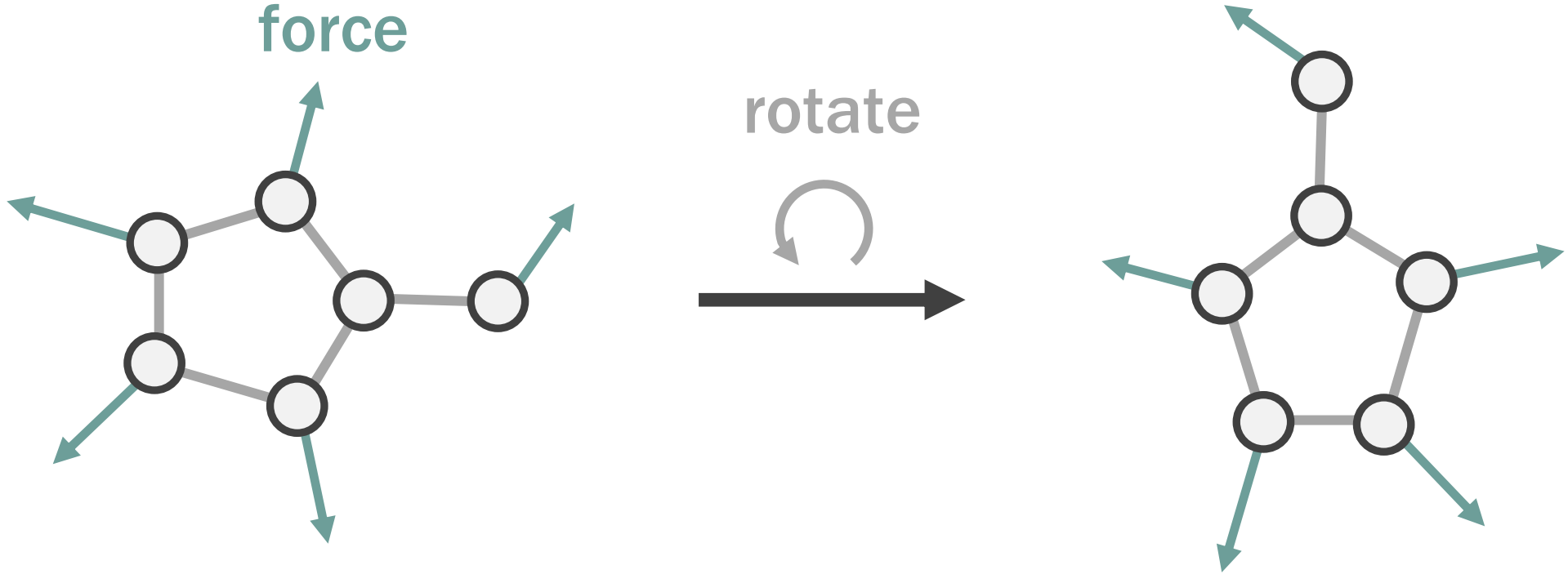


M3GNet

Bond angles and dihedrals

Vector and tensor properties — equivariance

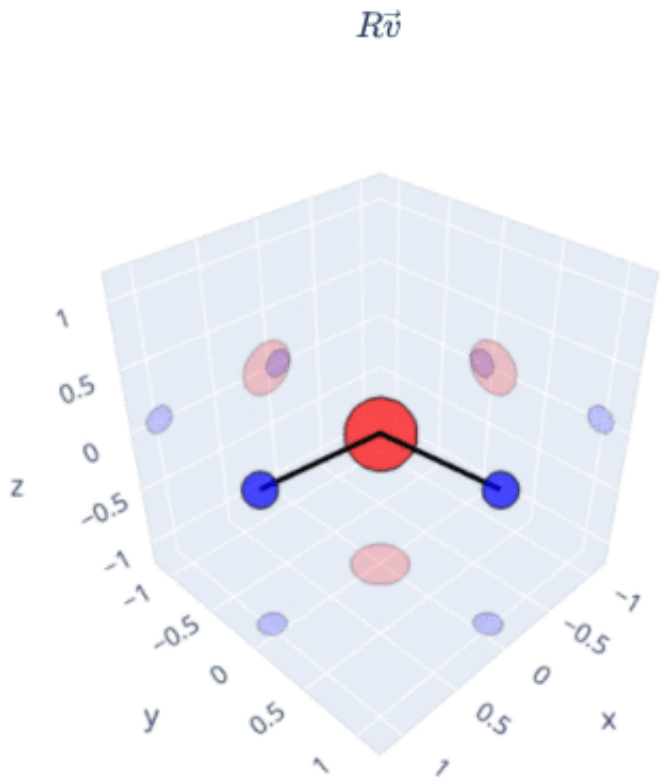
Higher dimensionality properties (vectors, tensors) such as force and stress require equivariant models

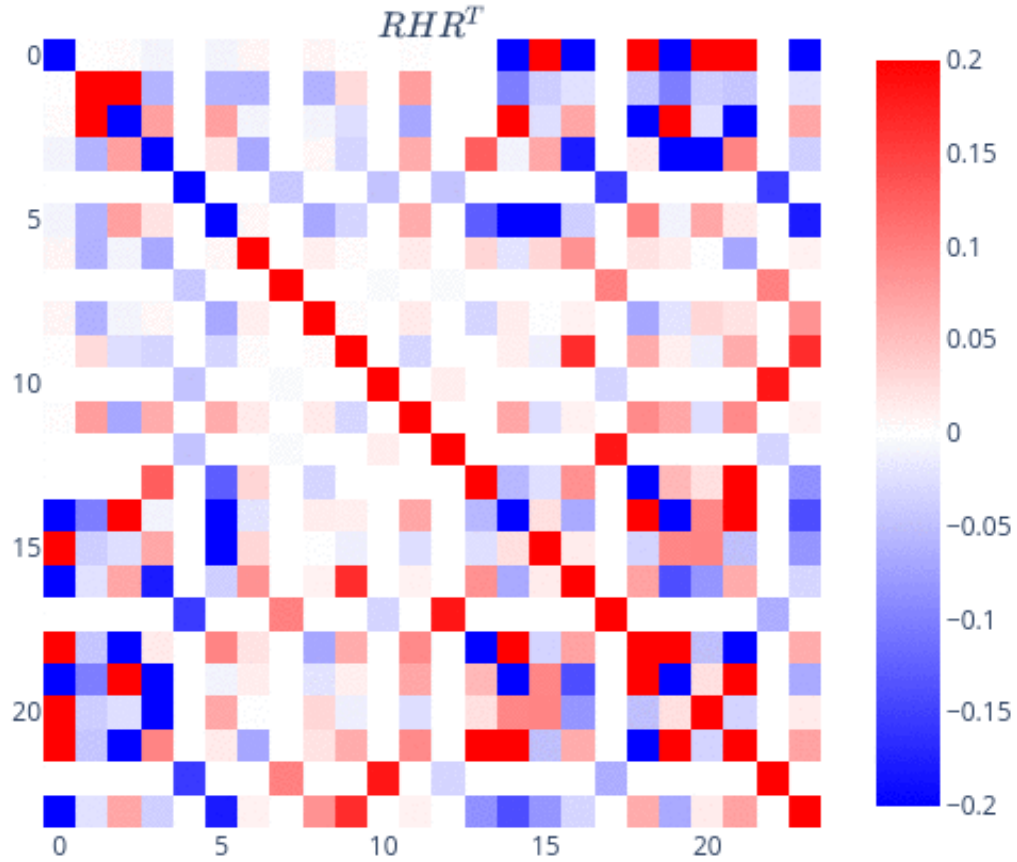


Forces should transform commensurate with the structure

Equivariant features

This requires features that transform predictably under rotations

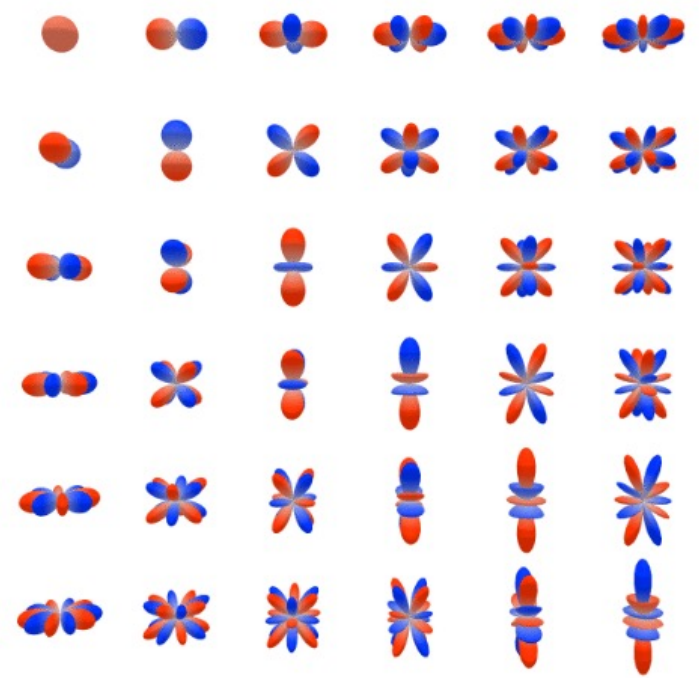




Credit: Tess Smidt, e3nn.org/e3nn-tutorial-mrs-fall-2021

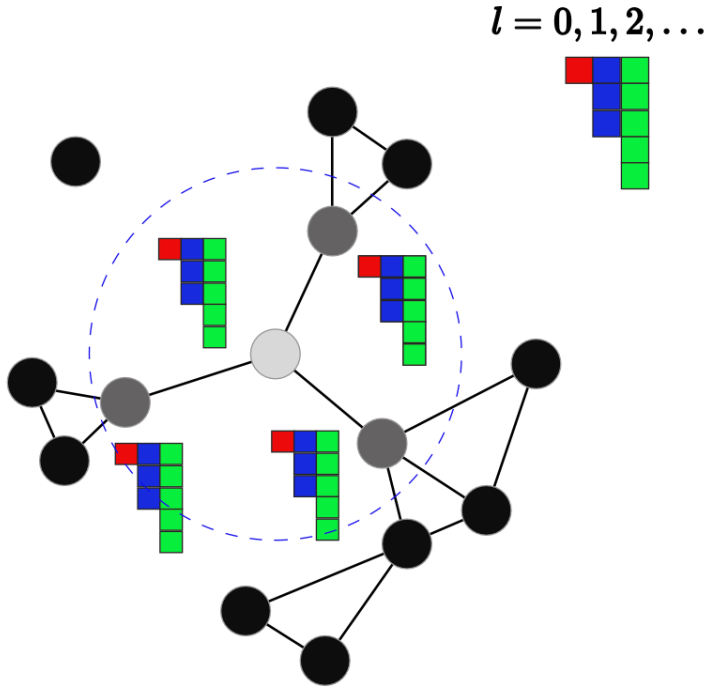
Equivariant graph models

Higher dimensionality properties (vectors, tensors) such as forces and stresses require equivariant models



e3nn

High-order spherical harmonic basis



Nequip

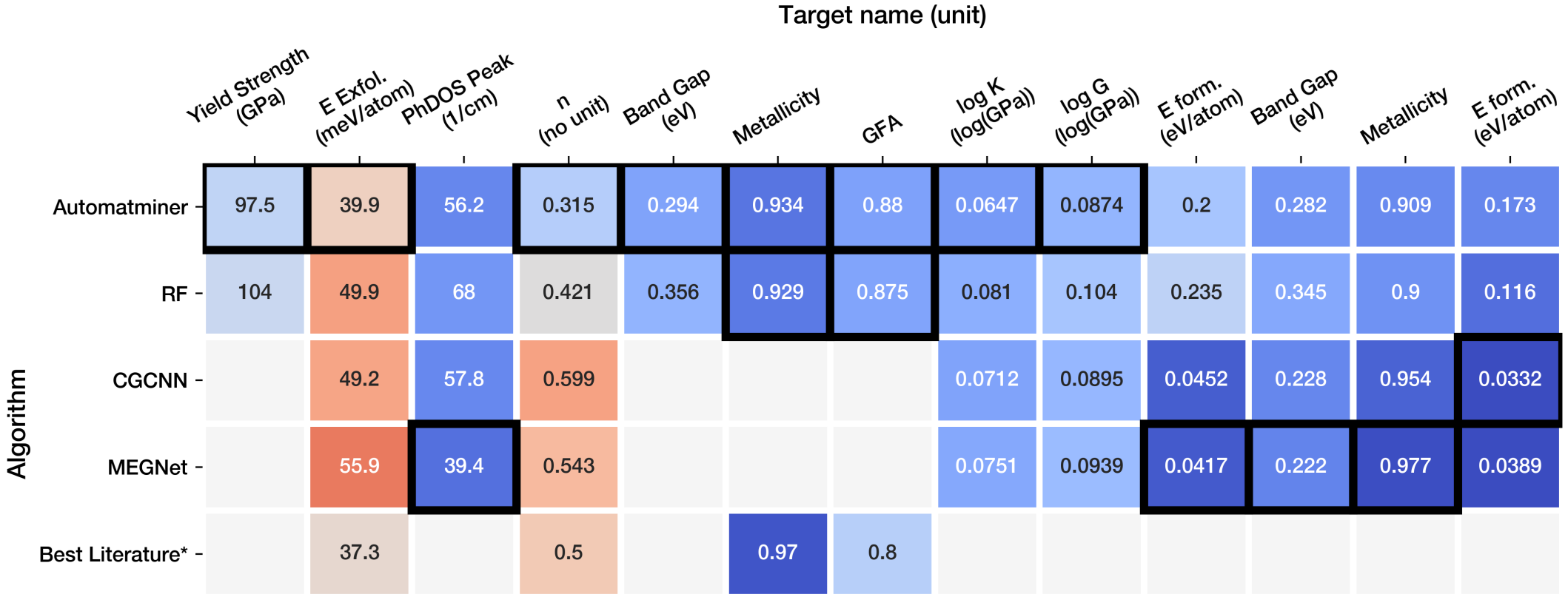
MLIP tensorial features

A large number of graph networks exist

	Gap meV	Homo meV	Lumo meV	C_V cal/mol K	μ D	ZPVE meV	R^2 α_0^2	α α_0^3	G meV	H meV	U meV	U_0 meV
NMP [68]	69	43	38	0.040	0.030	1.50	0.180	0.092	19	17	20	20
SchNet [13]	63	41	34	0.033	0.033	1.70	0.073	0.235	14	14	19	14
Cormorant [21]	61	34	38	0.026	0.038	2.03	0.961	0.085	20	21	21	22
LieConv [69]	49	30	25	0.038	0.032	2.28	0.800	0.084	22	24	19	19
DimeNet++ [14]	33	25	20	0.023	0.030	1.21	0.331	0.044	7.6	6.5	6.3	6.3
EGNN [70]	48	29	25	0.031	0.029	1.55	0.106	0.071	12	12	12	11
PaiNN [15]	46	28	20	0.024	0.012	1.28	0.066	0.045	7.4	6.0	5.8	5.9
TorchMD-NET [17]	36	20	18	0.026	**0.011**	1.84	**0.033**	0.059	7.6	6.2	6.4	6.2
SphereNet [71]	32	23	18	0.022	0.026	1.12	0.292	0.046	7.8	6.3	6.4	6.3
SEGNN [72]	42	24	21	0.031	0.023	1.62	0.660	0.060	15	16	13	15
EQGAT [73]	32	20	16	0.024	**0.011**	2.00	0.382	0.053	23	24	25	25
Equiformer [74]	30	**15**	**14**	0.023	**0.011**	1.26	0.251	0.046	7.6	6.6	6.7	6.6
MGCN [75]	64	42	57	0.038	0.056	1.12	0.110	**0.030**	15	16	14	13
Allegro [30]	-	-	-	-	-	-	-	-	5.7	4.4	4.4	4.7
NoisyNodes [76]	29	20	19	0.025	0.025	1.16	0.700	0.052	8.3	7.4	7.6	7.3
GNS-TAT+NN [77]	**26**	17	17	0.022	0.021	**1.08**	0.65	0.047	7.4	6.4	6.4	6.4
Wigner Kernels [78]	-	-	-	-	-	-	-	-	-	-	-	4.3
TensorNet [41]	-	-	-	-	-	-	-	-	6.0	**4.3**	4.3	4.3
MACE	42	22	19	**0.021**	0.015	1.23	0.210	0.038	**5.5**	4.7	**4.1**	**4.1**

Graph networks and the MatBench dataset

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



npj Comput. Mater. 6, 138 (2020)

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)

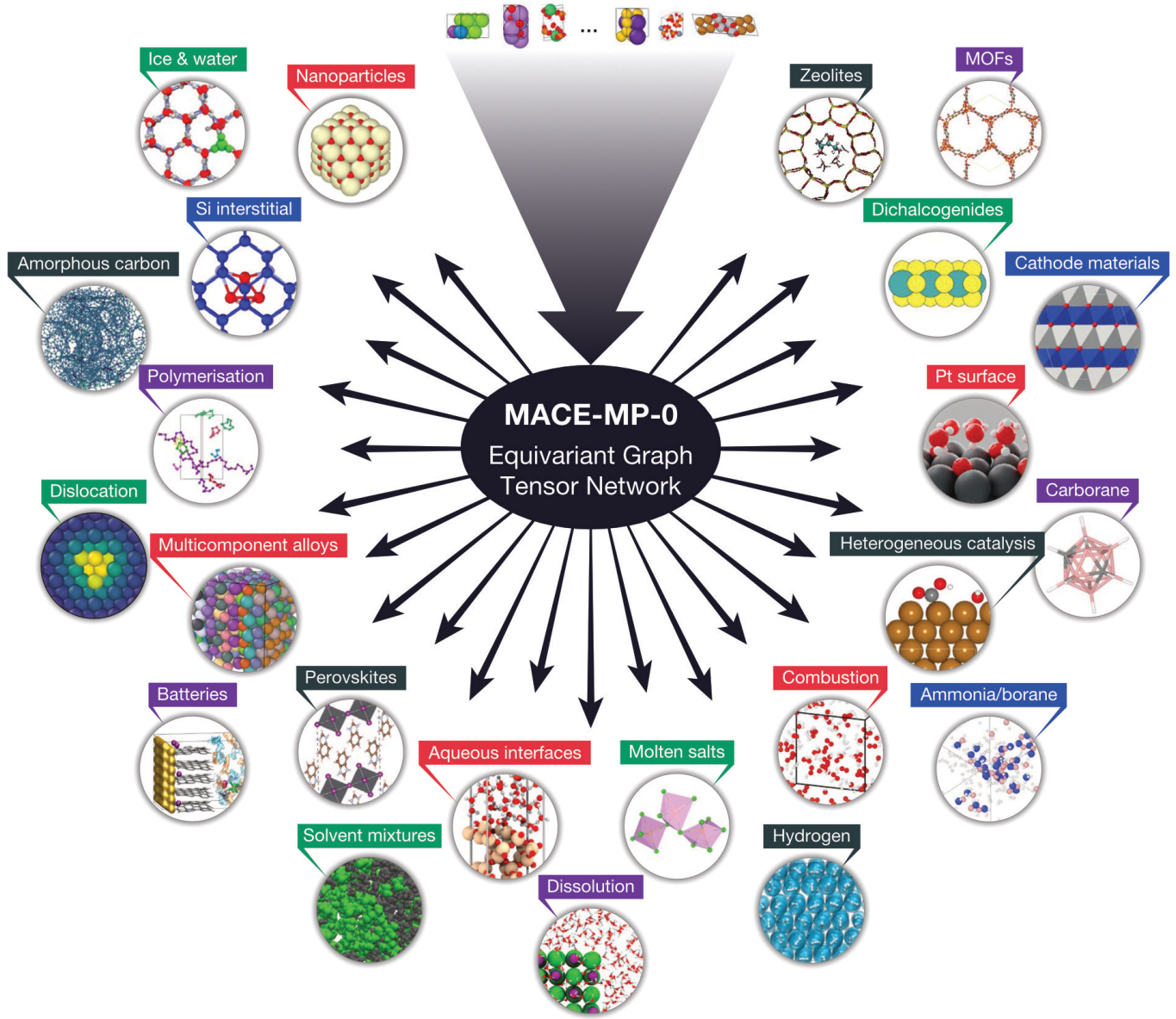
Task name	Samples	Algorithm	Verified MAE (unit) or ROCAUC	Notes
matbench_steels	312	MODNet (v0.1.12)	**87.7627 (MPa)**	
matbench_jdft2d	636	MODNet (v0.1.12)	**33.1918 (meV/atom)**	
matbench_phonons	1,265	MegNet (kgcnn v2.1.0)	**28.7606 (cm^-1)**	structure required
matbench_expt_gap	4,604	MODNet (v0.1.12)	**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

https://matbench.materialsproject.org

Universal force fields

Universal forcefields are an emerging paradigm in computational chemistry

Can be applied across the periodic table to predict energies, forces, and stresses



Matbench discovery leaderboard

Increasing:

- Accuracy
- Parameters
- Training data
- Equivariance

Model ⓘ	CPS ↑	F1 ↑	DAF ↑	Prec ↑	Acc ↑	MAE ↓	R^2 ↑	κ_{SRME} ↓	RMSD ↓	Training Set	Params
eSEN-30M-OAM	0.896	0.925	6.069	0.928	0.977	0.018	0.866	0.1704	0.0096	6.6M (113M) (OMat24+MPtrj+sAlex)	30.2M
SevenNet-MF-ompa	0.849	0.901	5.825	0.89	0.969	0.021	0.867	0.317	0.0115	6.6M (113M) (OMat24+sAlex+MPtrj)	25.7M
GRACE-2L-OAM	0.841	0.88	5.774	0.883	0.963	0.023	0.862	0.294	0.0121	6.6M (113M) (OMat24+sAlex+MPtrj)	12.6M
eSEN-30M-MP	0.800	0.831	5.26	0.804	0.946	0.033	0.822	0.3398	0.0142	146k (1.58M) (MPtrj)	30.1M
MACE-MPA-0	0.796	0.852	5.582	0.853	0.954	0.028	0.842	0.412	0.0143	3.37M (12M) (MPtrj+sAlex)	9.06M
DPA3-v2-OpenLAM	0.765	0.89	5.747	0.879	0.966	0.022	0.869	0.687	0.0127	163M (OpenLAM)	7.02M
GRACE-1L-OAM	0.762	0.824	5.255	0.803	0.944	0.031	0.842	0.516	0.0139	6.6M (113M) (OMat24+sAlex+MPtrj)	3.45M
MatterSim v1 5M	0.716	0.862	5.852	0.895	0.959	0.024	0.863	0.574	0.0733	17M (MatterSim)	4.55M
SevenNet-l3i5	0.709	0.76	4.629	0.708	0.92	0.044	0.776	0.55	0.0182	146k (1.58M) (MPtrj)	1.17M
MatRIS v0.5.0 MPtrj	0.680	0.809	5.049	0.772	0.938	0.037	0.803	0.861	0.0156	146k (1.58M) (MPtrj)	5.83M
GRACE-2L-MPtrj	0.678	0.691	4.163	0.636	0.896	0.052	0.741	0.525	0.0186	146k (1.58M) (MPtrj)	15.3M
DPA3-v2-MPtrj	0.647	0.786	4.822	0.737	0.929	0.039	0.804	0.959	0.0164	146k (1.58M) (MPtrj)	4.92M
MACE-MP-0	0.640	0.669	3.777	0.577	0.878	0.057	0.697	0.647	0.0194	146k (1.58M) (MPtrj)	4.69M
AlphaNet-MPTrj	0.562	0.799	4.863	0.743	0.933	0.041	0.745	1.31	0.0227	146k (1.58M) (MPtrj)	16.2M
eqV2 M	0.558	0.917	6.047	0.924	0.975	0.02	0.848	1.771	0.0138	3.37M (102M) (OMat24+MPtrj)	86.6M
ORB v2	0.540	0.88	6.041	0.924	0.965	0.028	0.824	1.732	0.016	3.25M (32.1M) (MPtrj+Alex)	25.2M
eqV2 S DeNS	0.526	0.815	5.042	0.771	0.941	0.036	0.788	1.676	0.0138	146k (1.58M) (MPtrj)	31.2M
ORB v2 MPtrj	0.476	0.765	4.702	0.719	0.922	0.045	0.756	1.725	0.0185	146k (1.58M) (MPtrj)	25.2M
M3GNet	0.430	0.569	2.882	0.441	0.813	0.075	0.585	1.412	0.0217	62.8k (188k) (MPF)	228k
CHGNet	0.391	0.613	3.361	0.514	0.851	0.063	0.689	1.717	0.0216	146k (1.58M) (MPtrj)	413k
GNoME	NaN	0.829	5.523	0.844	0.955	0.035	0.785	n/a	n/a	6M (89M) (GNoME)	16.2M

https://matbench-discovery.materialsproject.org

Summary

- 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