KAIST CHEMISTRY

# Graph neural network
# &
# Message passing neural network

Prof. Ph.D. Woo Youn Kim

Chemistry, KAIST

# Goals

| | | |
|---|---|---|
| 9주 | 주제 | Recurrent Neural Network (RNN) |
| | 목표 | Understanding RNN and molecular representation with smiles |
| | 내용 | RNN, LSTM, GRU Feature extraction of molecules using RNN |
| 10주 | 주제 | Message Passing Neural Network (MPNN) |
| | 목표 | Understanding the most general expression of graph neural network |
| | 내용 | MPNN, molecular graph representation, GGNN,<br>supervised learning of logP and TPSA |
| 11주 | 주제 | Molecular generative model 1 |
| | 목표 | Understanding the principle of autoencoder and unsupervised learning |
| | 내용 | Molecular autoencoder, VAE, CVAE, *de novo* molecular design |

# Review

Bayes' Rule

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} = \frac{P(x|\theta)P(\theta)}{\sum_{\theta} P(x|\theta)P(\theta)}$$

$$\text{Posterior probability} = \frac{(\text{likelihood}) \times (\text{prior probability})}{(\text{evidence})}$$

Prior probability (사전확률) $P(\theta)$: probability of a parameter set $\theta$.

Posterior probability (사후확률) $P(\theta|x)$: $P(\theta)$ given an observation $x$.

Evidence (증거) $P(x)$: probability of the observation.

Likelihood (가능도) $L(\theta|x) = P(x|\theta)$: $P(x)$ given the parameters $\theta$

$\theta$ = disease

$x$ = test result (T or F)

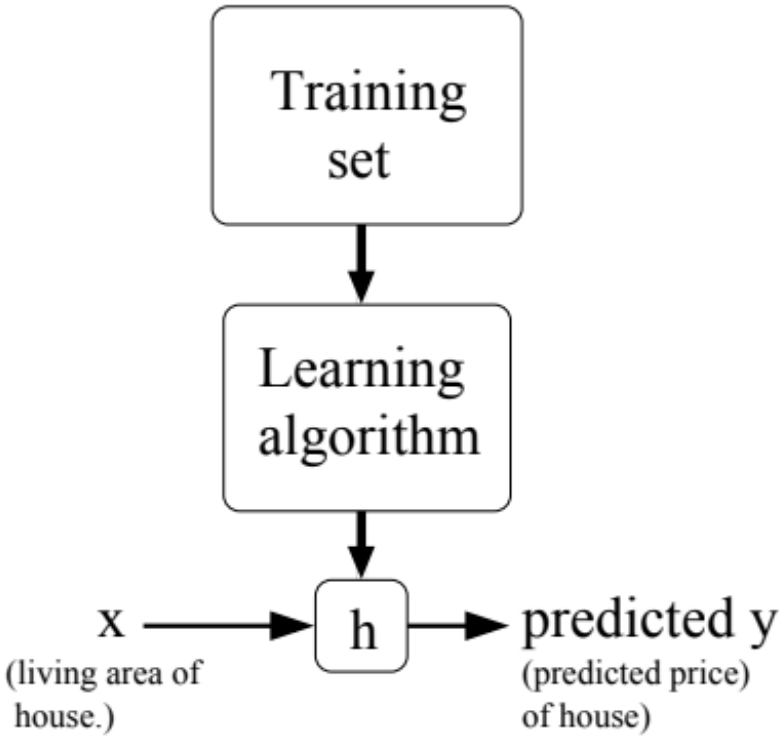
$P(x)$: probability of T and F

$P(\theta)$: probability of having a disease

$P(\theta|x)$: probability of having the disease given the test result

$P(x)$: probability of T and F.

# Review




The principal of **maximum likelihood** says that we should choose θ so as to make the data as high probability as possible.

# Review

Instead of maximizing L(θ), we can also maximize any strictly increasing function of L(θ).

For example, maximize the ***log likelihood*** $l(\theta)$:

$$\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2
\end{aligned}$$

Note that the final θ has no dependence on σ.

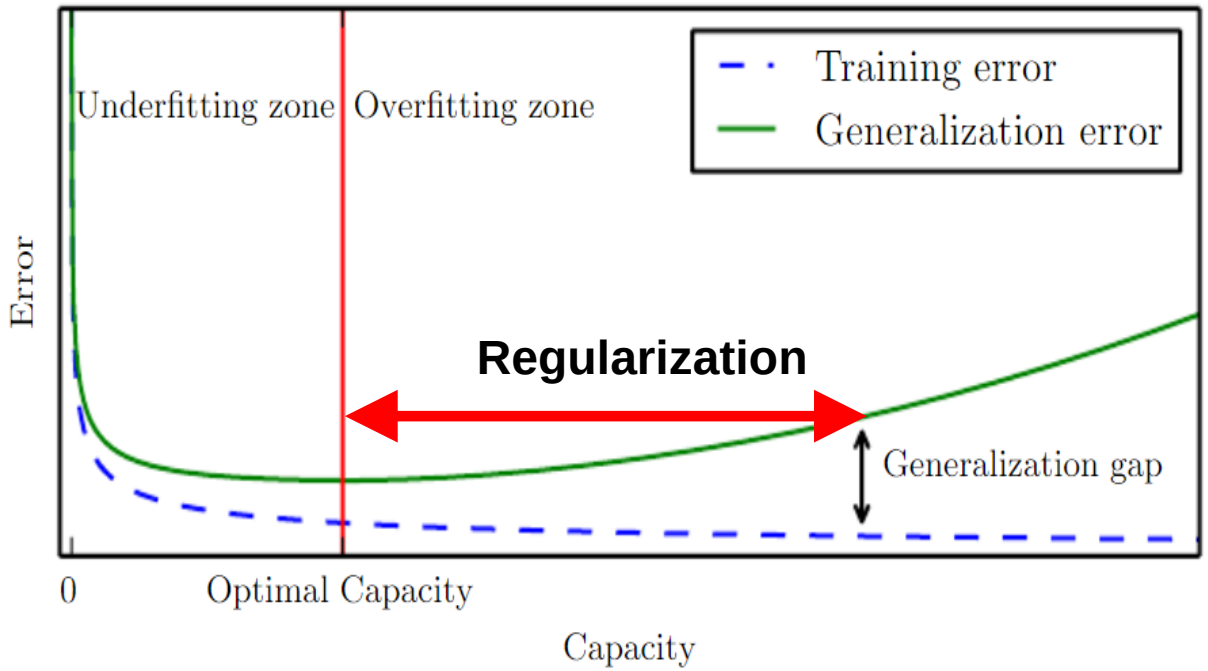
Maximizing $l$ (θ) gives the same answer as minimizing

$$\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

which is the original LMS cost function.

# Review

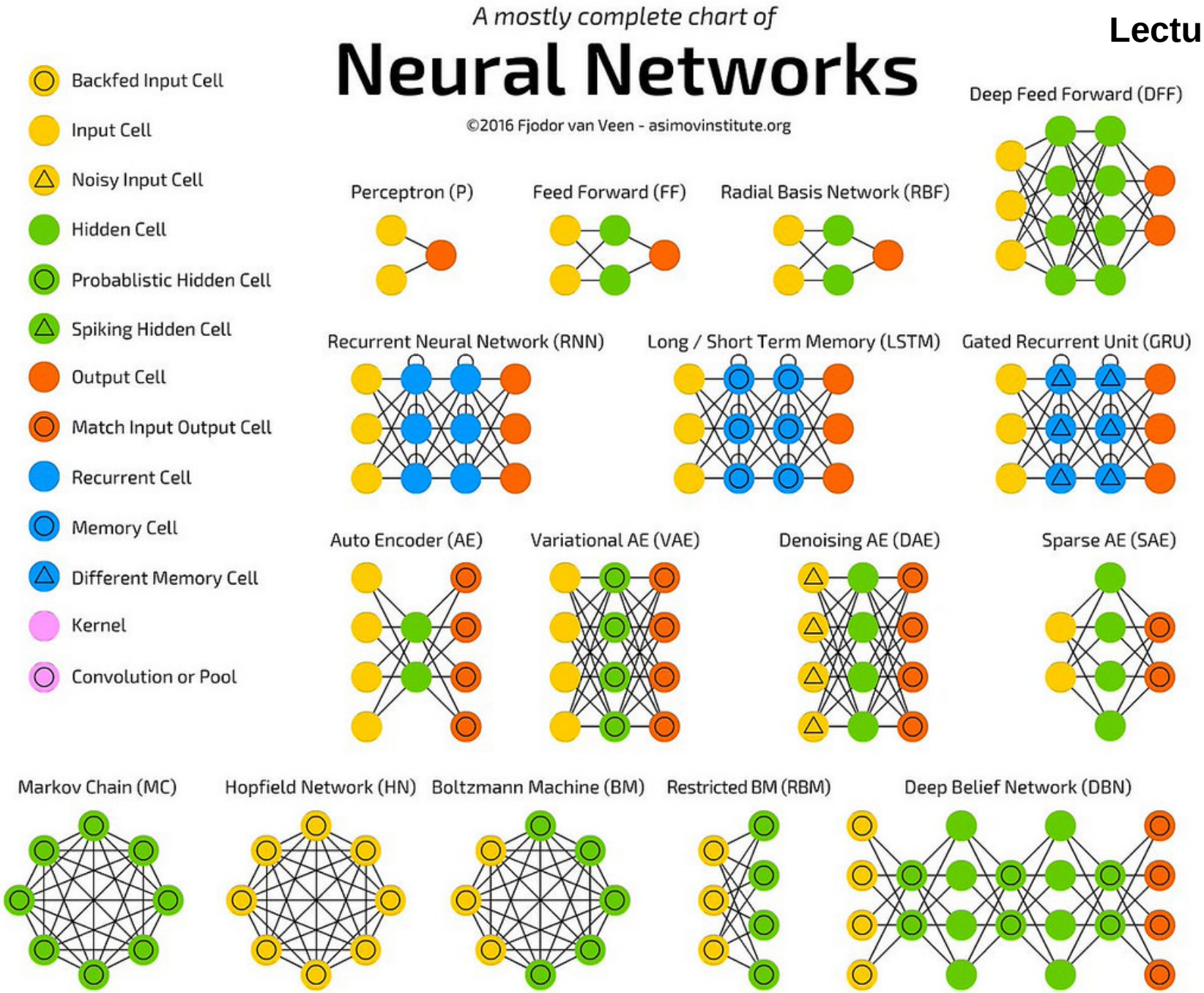
- The main challenge is to find an optimal capacity of model for a given task.
- **Regularization** is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.




1. **Data augmentation: Big data**
2. **Model selection via cross validation**
3. **L1,L2-Regularization**
4. **Dropout**

# Review

Choose a right model for a given problem to minimize generalization errors



# Reference paper

## Relational inductive biases, deep learning, and graph networks

**Peter W. Battaglia[1]*, Jessica B. Hamrick[1], Victor Bapst[1], Alvaro Sanchez-Gonzalez[1], Vinicius Zambaldi[1], Mateusz Malinowski[1], Andrea Tacchetti[1], David Raposo[1], Adam Santoro[1], Ryan Faulkner[1], Caglar Gulcehre[1], Francis Song[1], Andrew Ballard[1], Justin Gilmer[2], George Dahl[2], Ashish Vaswani[2], Kelsey Allen[3], Charles Nash[4], Victoria Langston[1], Chris Dyer[1], Nicolas Heess[1], Daan Wierstra[1], Pushmeet Kohli[1], Matt Botvinick[1], Oriol Vinyals[1], Yujia Li[1], Razvan Pascanu[1]**

[1]DeepMind; [2]Google Brain; [3]MIT; [4]University of Edinburgh

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

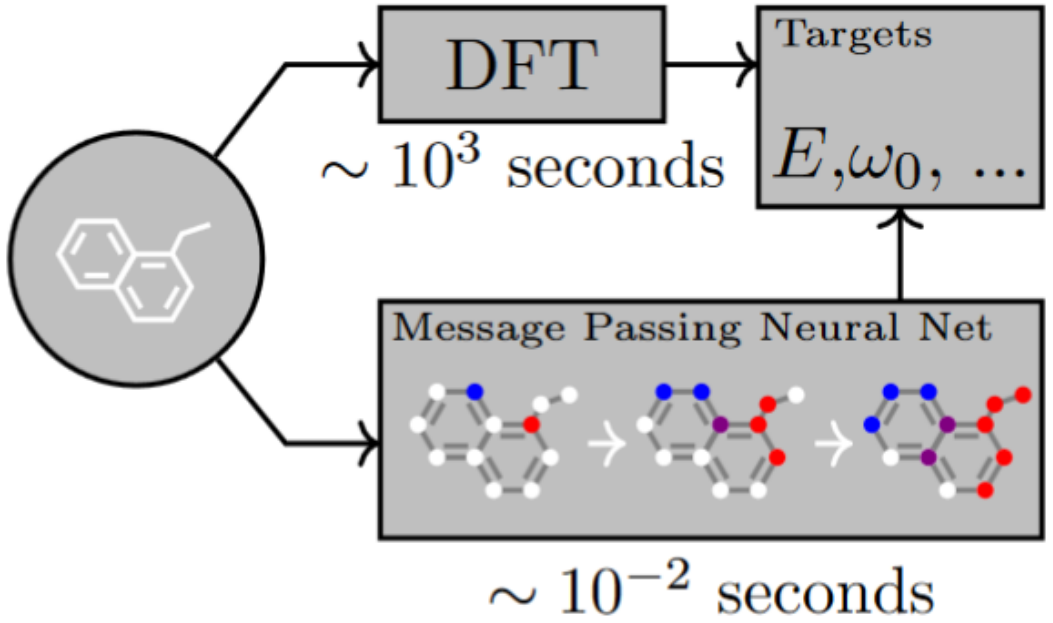
# Reference paper

## Neural Message Passing for Quantum Chemistry

**Justin Gilmer** [1] **Samuel S. Schoenholz** [1] **Patrick F. Riley** [2] **Oriol Vinyals** [3] **George E. Dahl** [1]

[1]Google Brain [2]Google [3]Google DeepMind. Correspondence to: Justin Gilmer <gilmer@google.com>, George E. Dahl <gdahl@google.com>.

*Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, PMLR 70, 2017. 




Gilmer, Justin, et al. "Neural message passing for quantum chemistry." *arXiv preprint arXiv:1704.01212* (2017).

# Contents

- Relational reasoning
- Inductive biases in neural network
- Graph neural network
- Message passing neural network
- MPNN for quantum chemistry

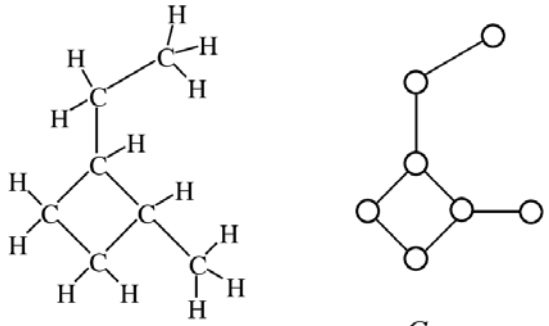
KAIST

# Relational reasoning

# Relational reasoning

**Important notions**

- **Structure** as the product of composing a set of known building blocks.
  - ✓ “Structured representations” capture this composition (i.e., the arrangement of the elements)
  - ✓ “structured computations” operate over the elements and their composition as a whole.



$G_1$

- **Entity** is an element with attributes, such as a physical object with a size and mass.
- **Relation** is a property between entities.
  - ✓ same size as, heavier than, and distance from
  - ✓ relations can have attributes as well: C=C, C-C, where bond orders are relational attributes

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Relational reasoning

**Important notions**

- **Structure** as the product of composing a set of known building blocks.
- **Entity** is an element with attributes, such as a physical object with a size and mass.
- **Relation** is a property between entities.
- **Rule** is a function that maps entities and relations to other entities and relations.
  - ✓ ex) a scale comparison like is entity X large? and is entity X heavier than entity Y?

Relational reasoning, then, involves manipulating **structured representations** of **entities** and **relations**, using **rules** for how they can be composed. We use these terms to capture notions from cognitive science, theoretical computer science, and AI

We explore how using **relational inductive biases** within deep learning architectures can facilitate learning about entities, relations, and rules for composing them

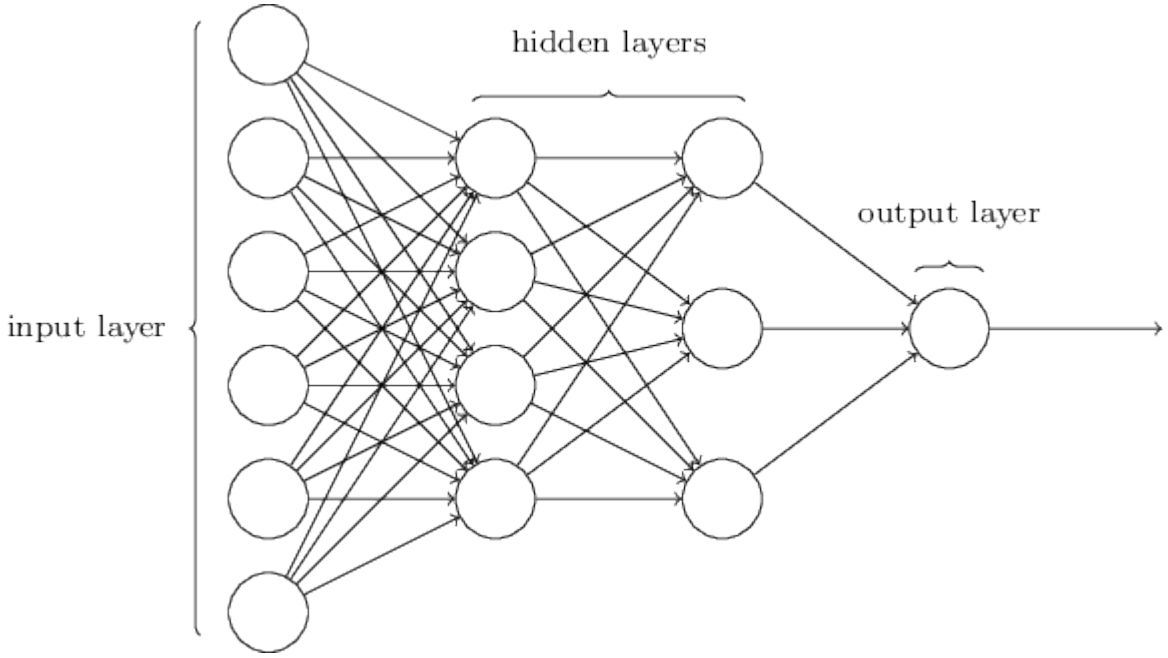
KAIST

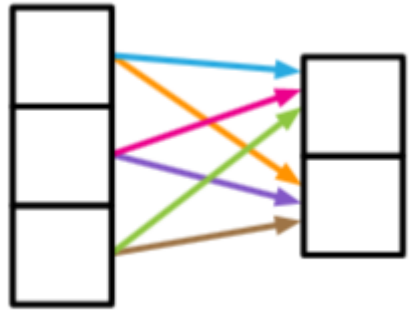
Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Inductive biases in neural network

# Weight sharing in neural network

**Fully-connected neural network (also referred to as multi-layer perceptron)**



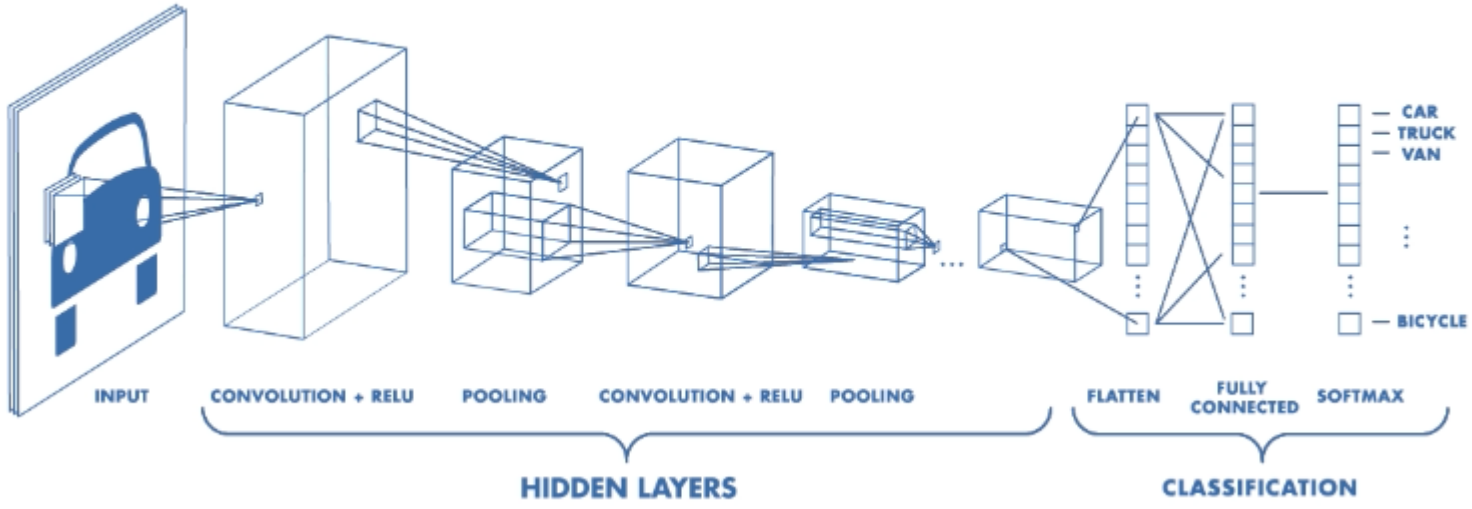



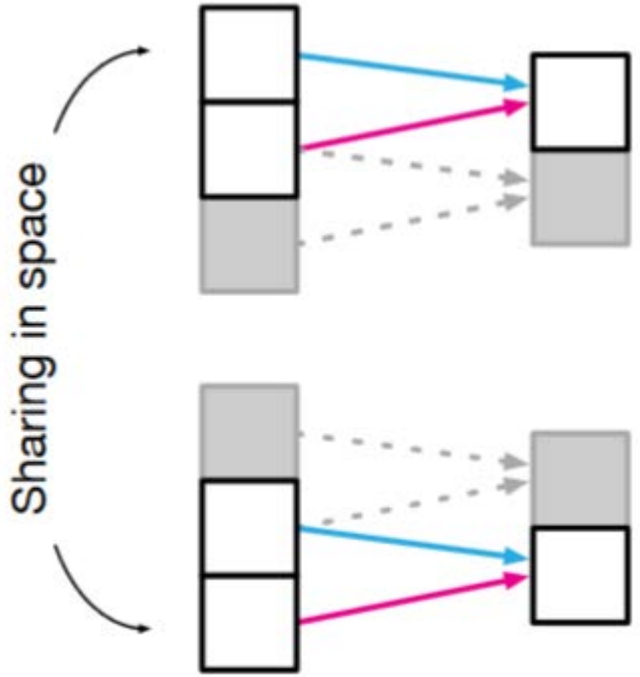
No weight sharing

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Weight sharing in neural network

**Convolutional neural network**

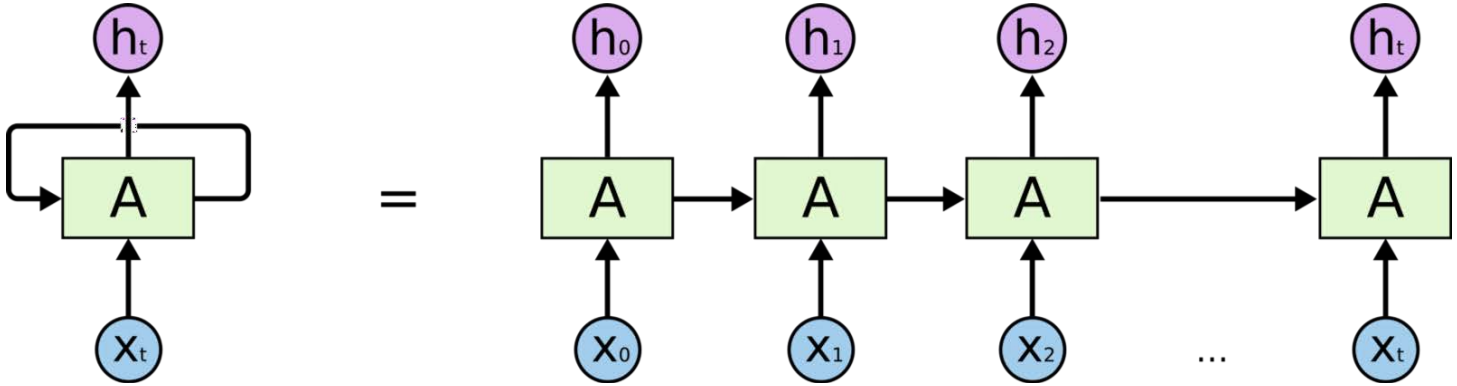


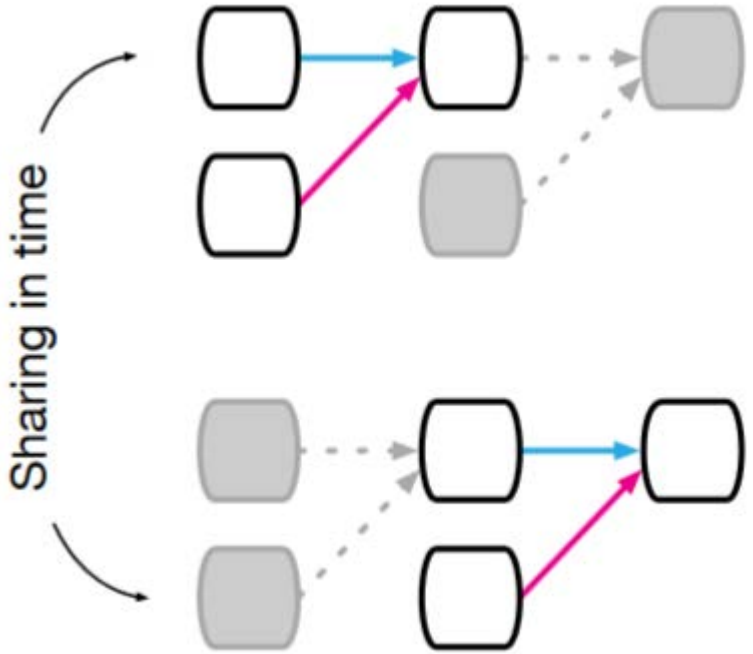




Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Weight sharing in neural network

**Recurrent neural network**

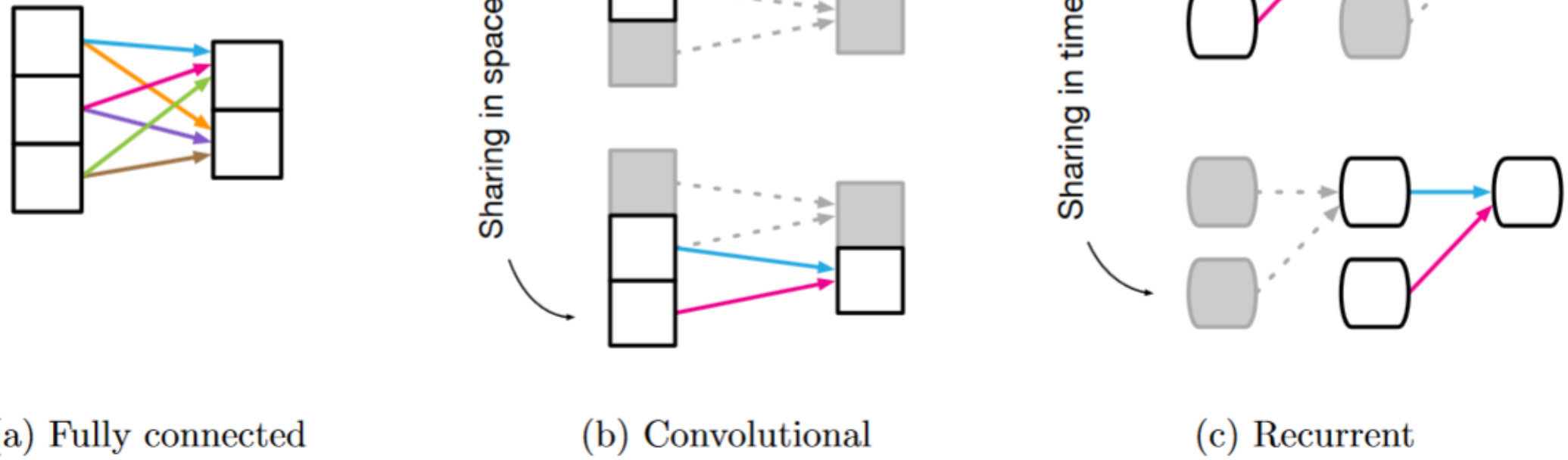



Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Weight sharing in neural network

**Inductive bias is another name of weight sharing**



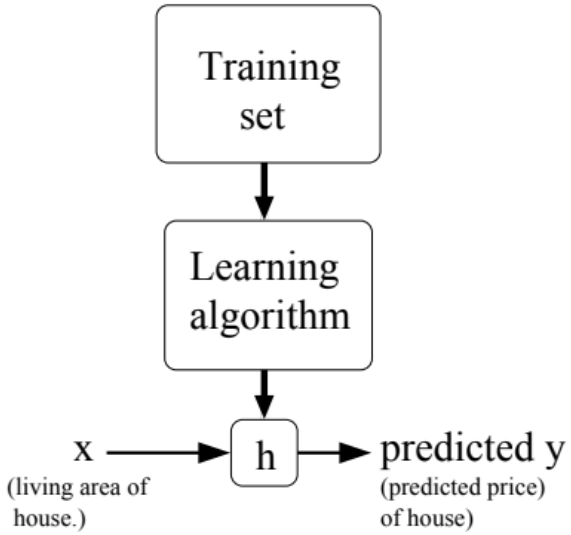

(a) Fully connected (b) Convolutional (c) Recurrent

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Inductive biases

**Interpretations of the inductive bias**

- ✓ An inductive bias allows a learning algorithm to prioritize one solution (or interpretation) over another, independent of the observed data.




The principal of **maximum likelihood** says that we should choose θ so as to make the data as high probability as possible.

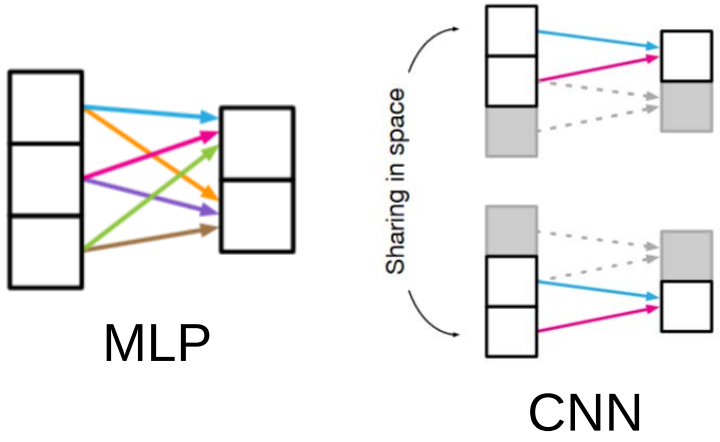
Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Inductive biases

**Interpretations of the inductive bias**

- ✓ An inductive bias allows a learning algorithm to prioritize one solution (or interpretation) over another, independent of the observed data.
- ✓ In a Bayesian model, inductive biases are typically expressed through the choice and parameterization of the prior distribution.

$$p(\omega|X,Y) = \frac{p(Y|X,\omega) \cdot \boldsymbol{p(\omega)}}{p(Y|X)}$$




MLP CNN



Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Inductive biases

**Interpretations of the inductive bias**

- ✓ An inductive bias allows a learning algorithm to prioritize one solution (or interpretation) over another, independent of the observed data.
- ✓ In a Bayesian model, the inductive bias is typically expressed through the choice and parameterization of the prior distribution.

$$p(\omega|X,Y) = \frac{p(Y|X,\omega) \cdot \boldsymbol{p(\omega)}}{p(Y|X)}$$

- ✓ In other contexts, the inductive bias might be a regularization term added to avoid overfitting, or it might be encoded in the architecture of the algorithm itself.



Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Inductive biases

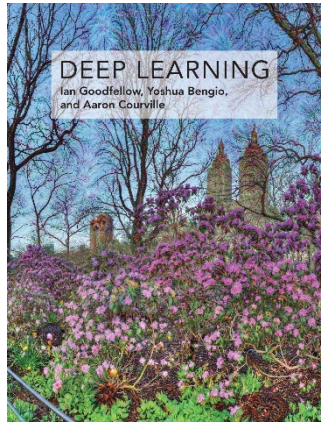
**Interpretations of the inductive bias**

**Priors can be considered weak or strong depending on how concentrated the probability density in the prior is**.

- **weak prior**: a prior distribution with high entropy, such as a Gaussian distribution with high variance.
  - ✓ such a prior allows the data to move the parameters more or less freely.

- **strong prior:** a prior distribution with very low entropy, such as a Gaussian distribution with low variance.
  - ✓ such a prior plays a more active role in determining where the parameters end up.

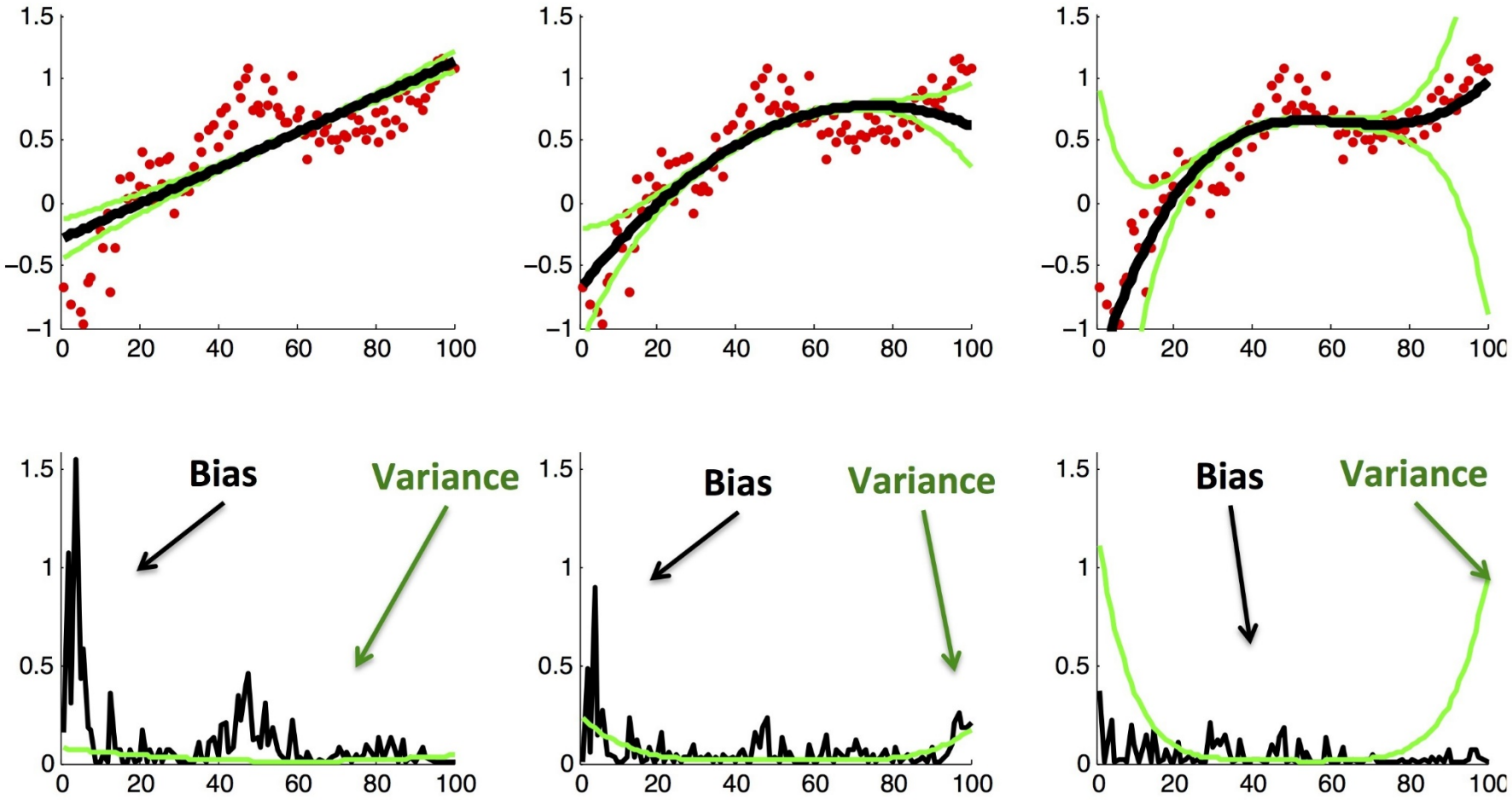
An infinitely strong prior places zero probability on some parameters and says that these parameter values are completely forbidden, regardless of how much support the data gives to those values.




- In 9.4 Convolution and Pooling as an Infinitely Strong Prior

Goodfellow, Ian, et al. *Deep learning*. Vol. 1. Cambridge: MIT press, 2016.

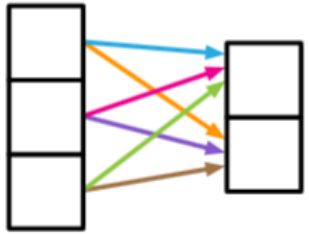
# Review: Bias-variance tradeoff

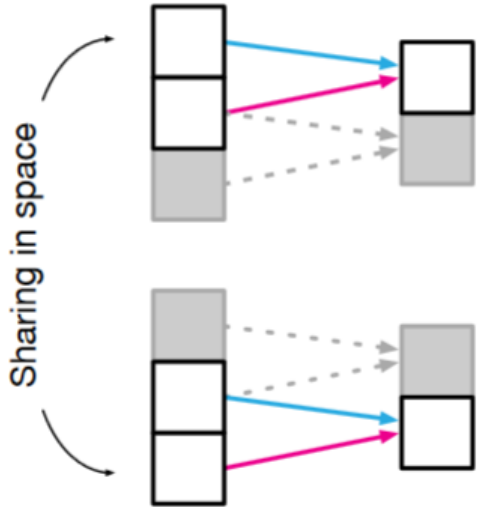


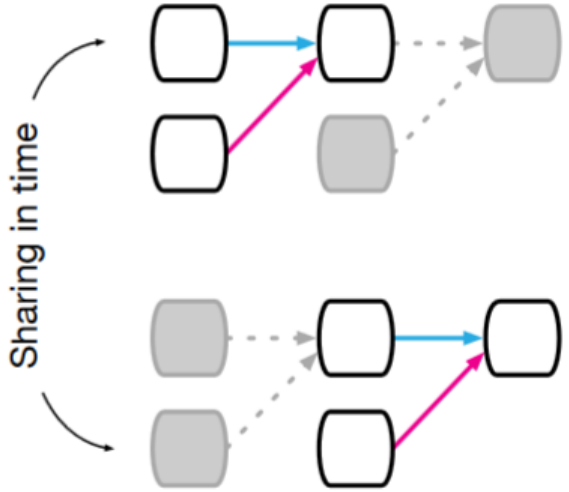

If you train on more, then the tradeoff will shift in favor of cubic models, because the bias term will dominate.

# Inductive biases

**Inductive biases in neural networks**





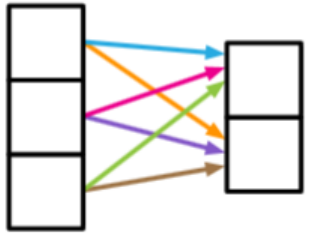



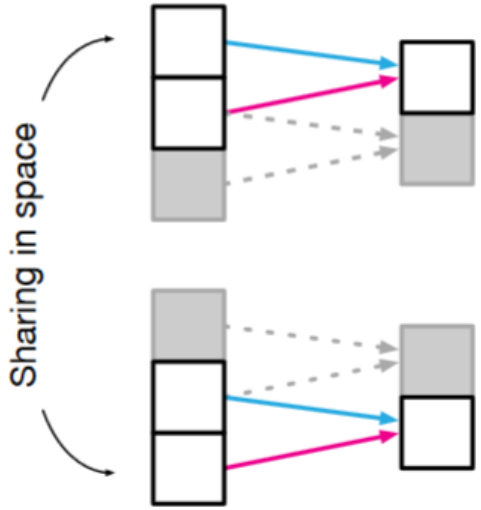

| Component | Entities | Relations | Rel. inductive bias | Invariance |
|---|---|---|---|---|
| Fully connected | Units | All-to-all | Weak | - |
| Convolutional | Grid elements | Local | Locality | Spatial translation |
| Recurrent | Timesteps | Sequential | Sequentiality | Time translation |

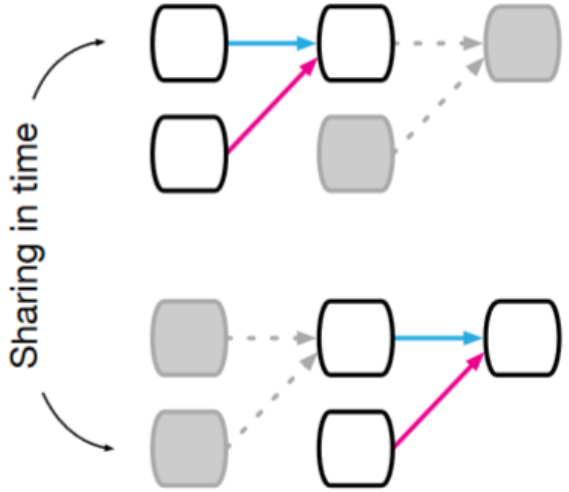
Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Inductive biases

**Inductive biases in neural networks**





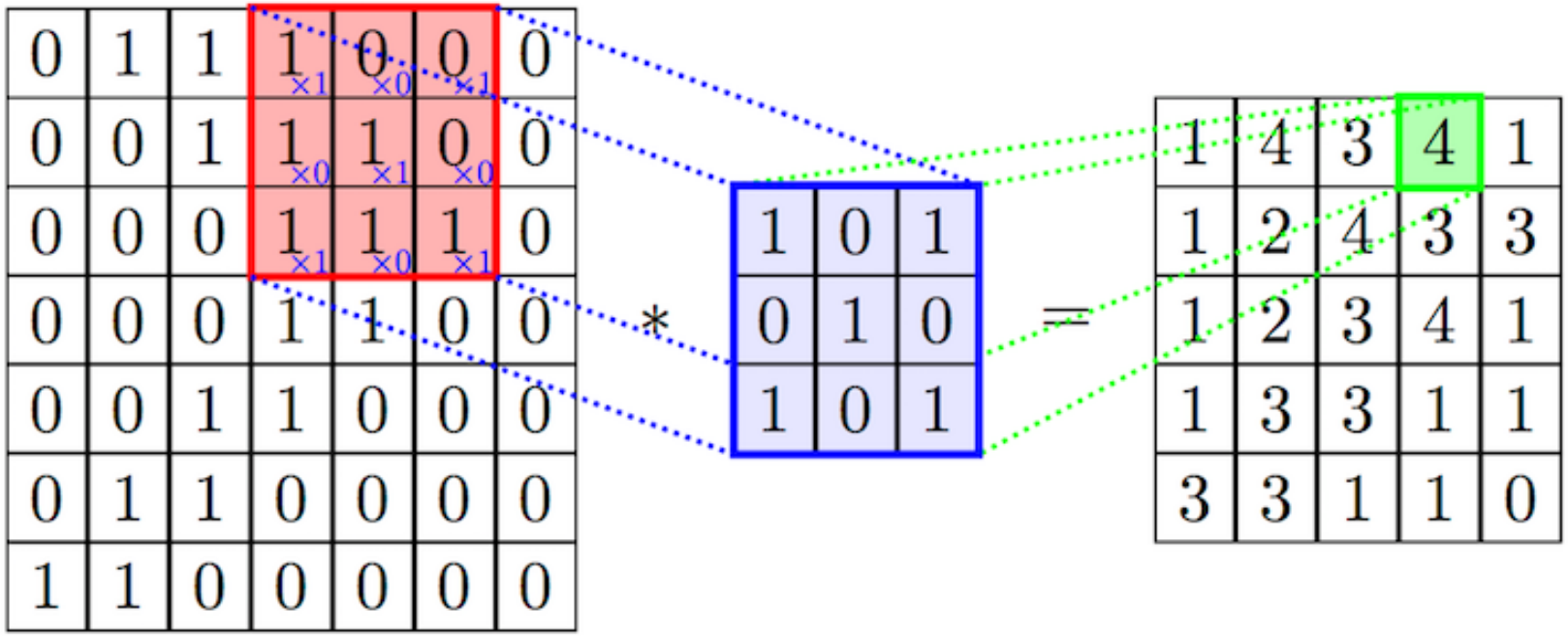




| Component | Entities | Relations | Rel. inductive bias | Invariance |
|---|---|---|---|---|
| Fully connected | Units | All-to-all | Weak | - |
| Convolutional | Grid elements | Local | Locality | Spatial translation |
| Recurrent | Timesteps | Sequential | Sequentiality | Time translation |

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Graph neural network

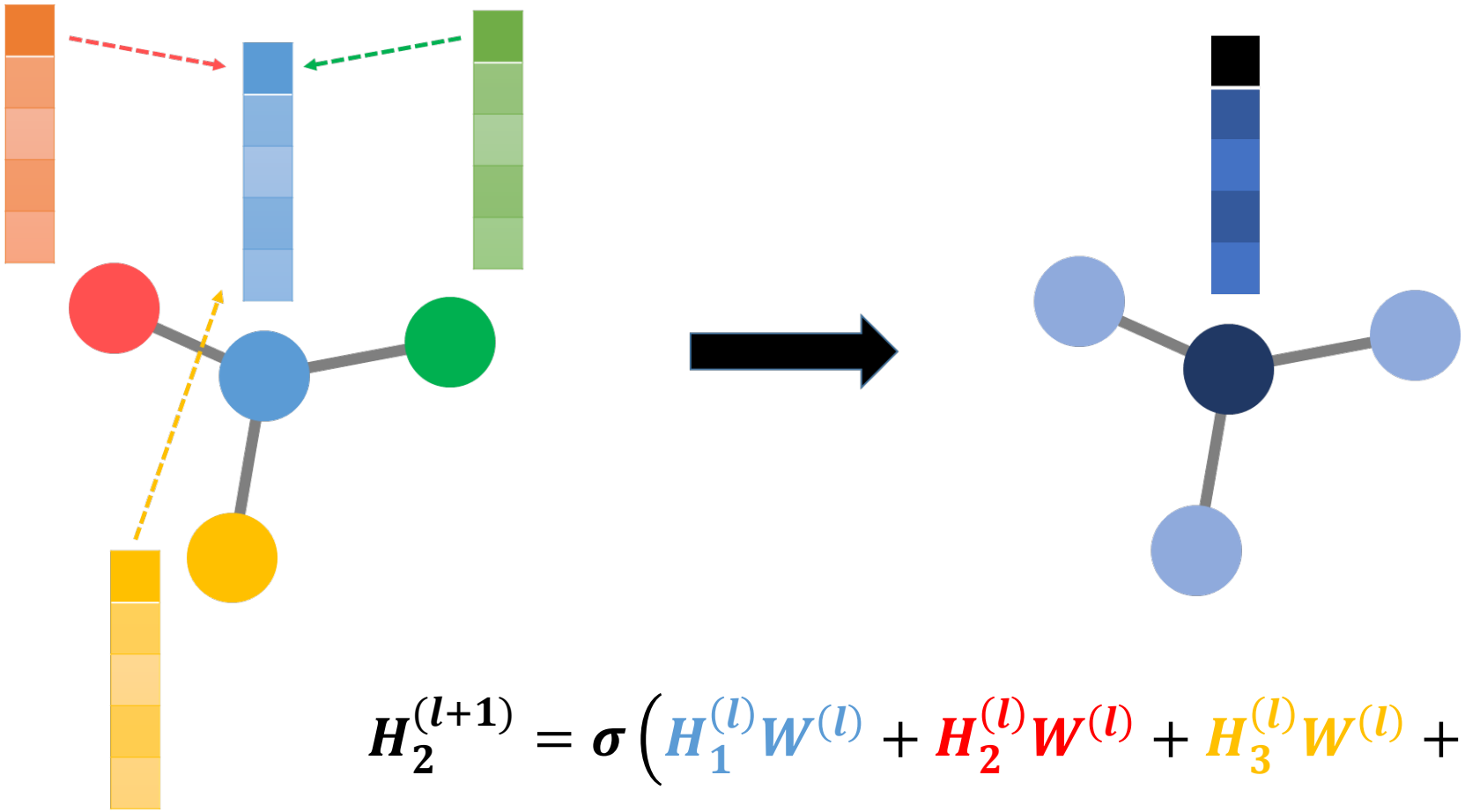
# Convolutional neural network




$$X_i^{(l+1)} = \sigma\left(\sum_{j\in[i-k,i+k]} W_j^{(l)} X_j^{(l)} + b^{(l)}\right)$$

Learnable parameters are shared

# Graph neural network

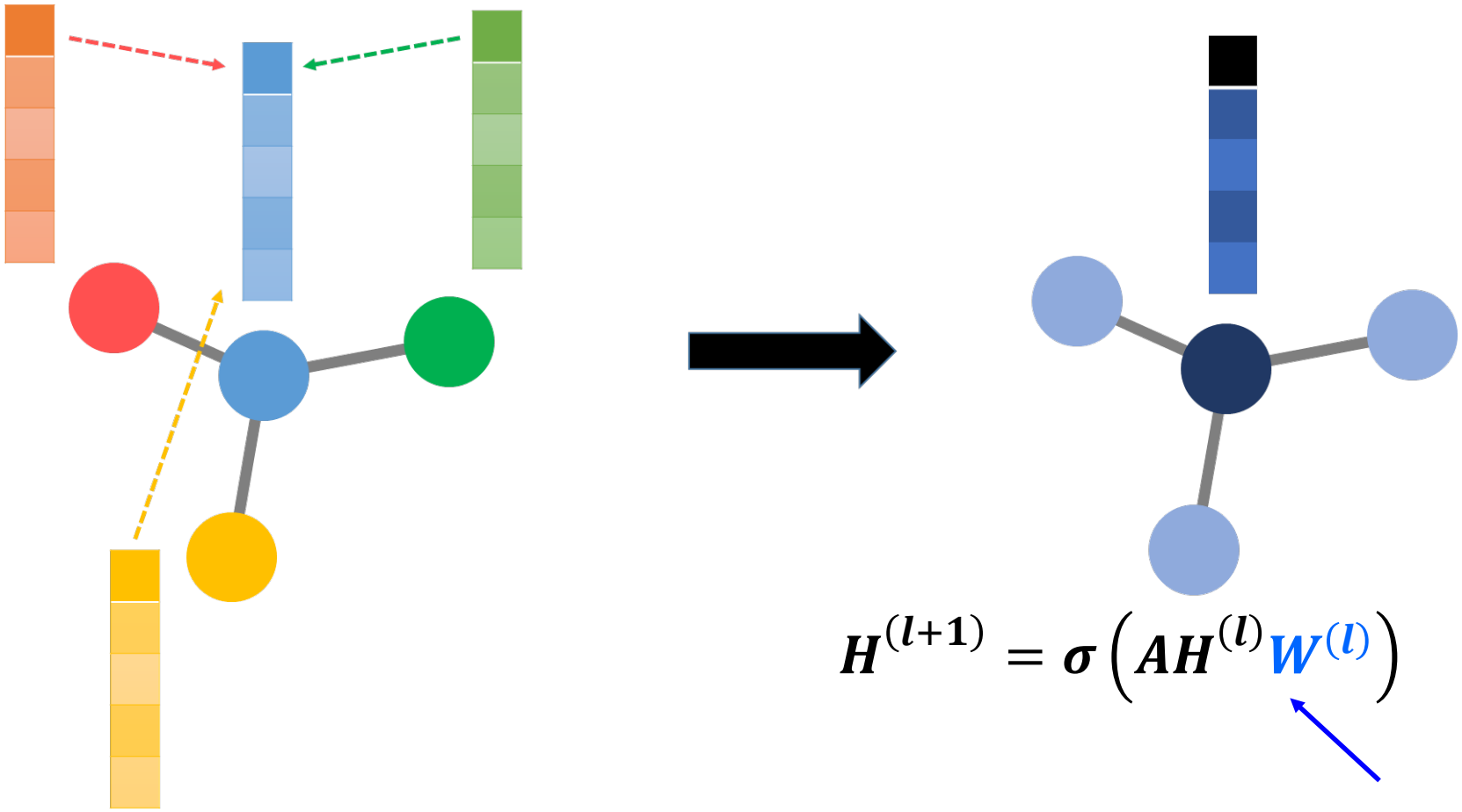


$$H_2^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)}\right)$$

$$\Rightarrow \quad H_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} H_j^{(l)}W^{(l)}\right)$$

KAIST

# Graph convolutional network

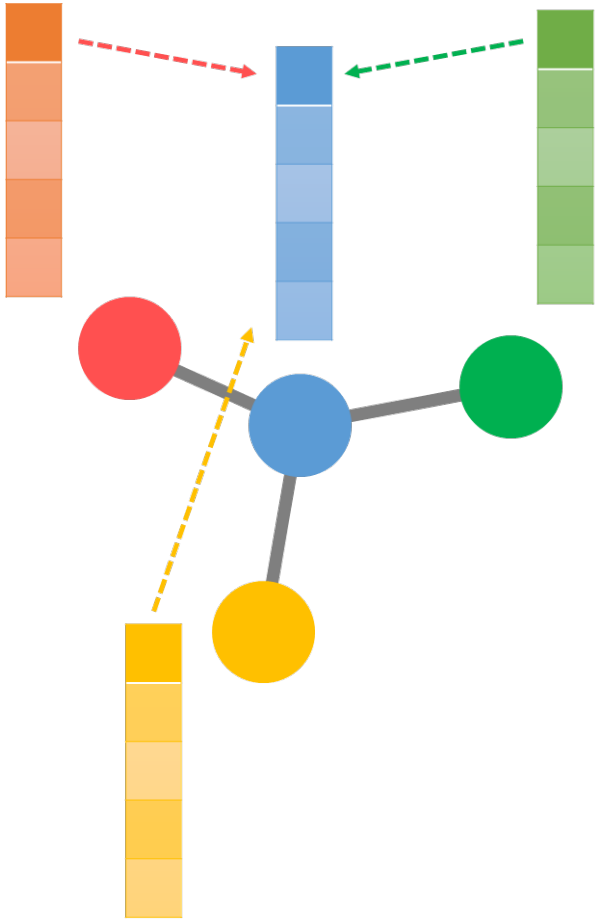


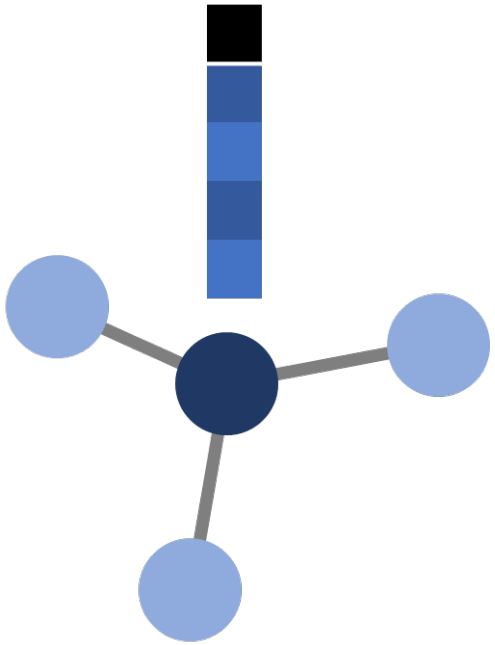
$$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)}\right)$$

Learnable parameter is shared

**Question) What is the inductive bias for GCN?**

# Graph convolutional network





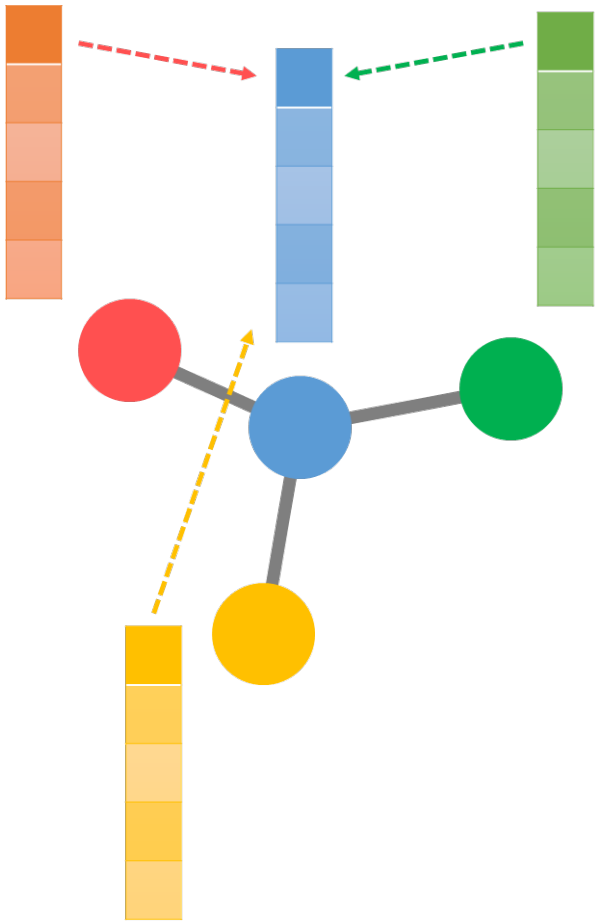
$$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)}\right)$$

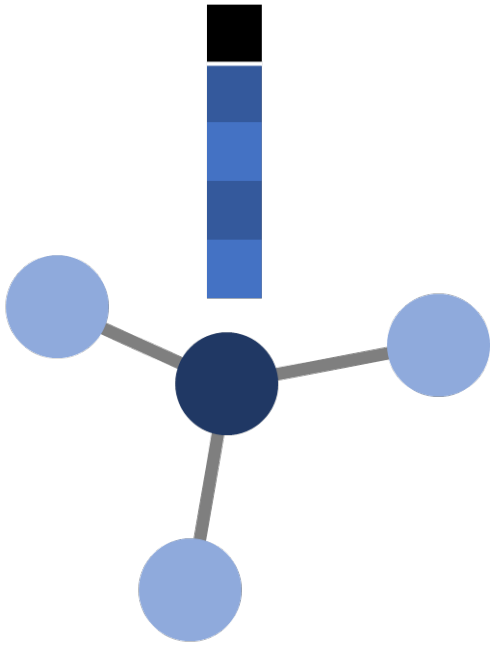
Learnable parameter is shared

**Question) What is the inductive bias for GCN?**

**Answer) Connectivity between nodes – the adjacency matrix**

# Graph convolutional network





$$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)}\right)$$

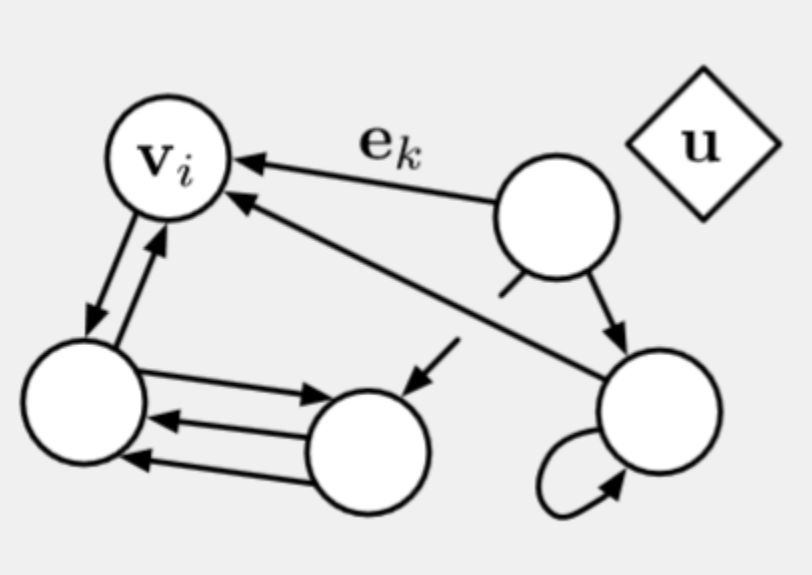
Learnable parameter is shared

**All nodes in graph share weights,**

**but nodes are differently updated by reflecting individual node features, $H_j^{(l)}$**

# Graph neural network

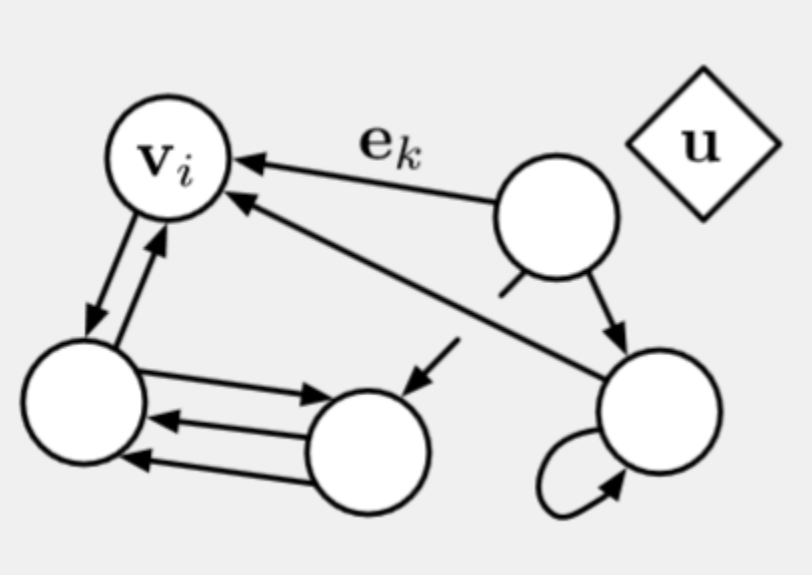
**Graph neural networks**




Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Graph neural network

**Graph neural networks**

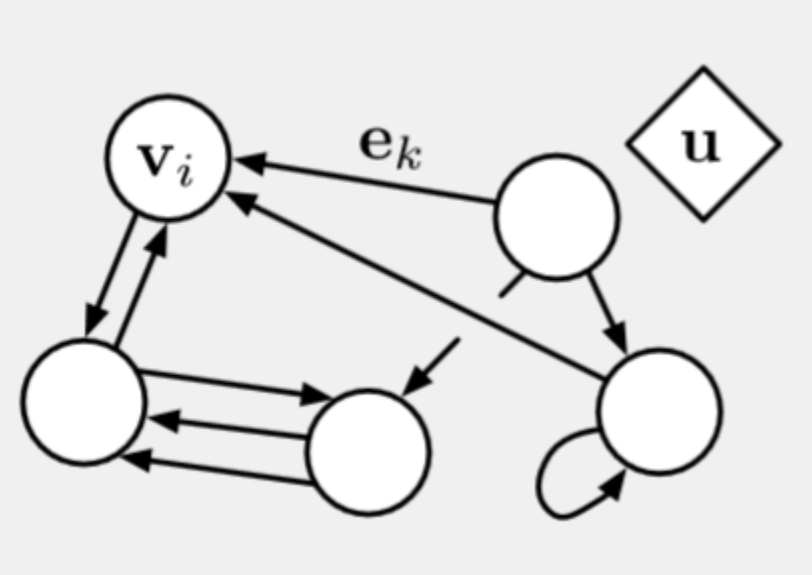



**Node's attribute (feature)**


Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

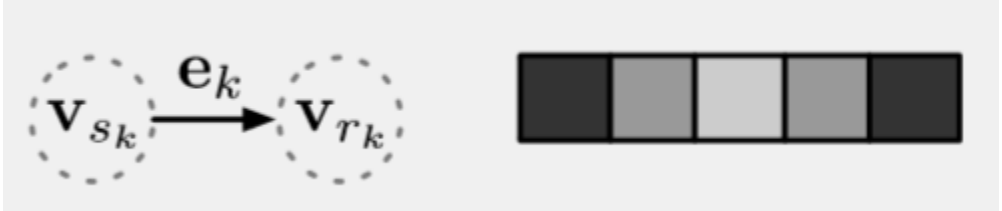
# Graph neural network

**Graph neural networks**




**Node's attribute (feature)**


**Edge's attribute (feature)**

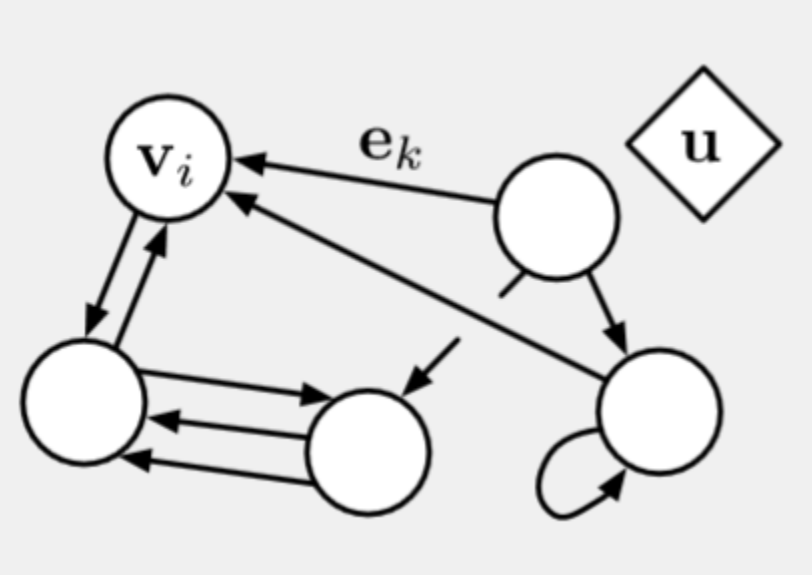



KAIST

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Graph neural network

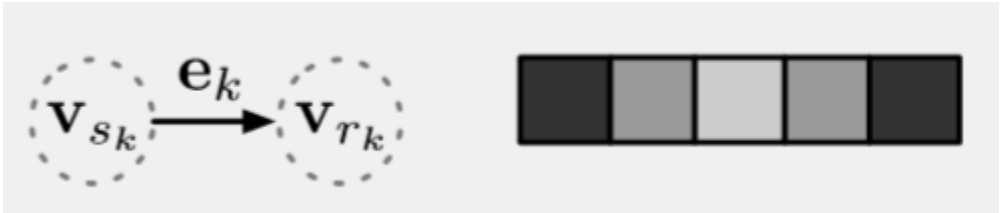
**Graph neural networks**




- ✓ Directed : one-way edges, from a “sender” node to a “receiver” node.
- ✓ Attribute : properties that can be encoded as a vector set, or even another graph
- ✓ Attributed : edges and vertices have attributes associated with them

**Node’s attribute (feature)**


**Edge’s attribute (feature)**



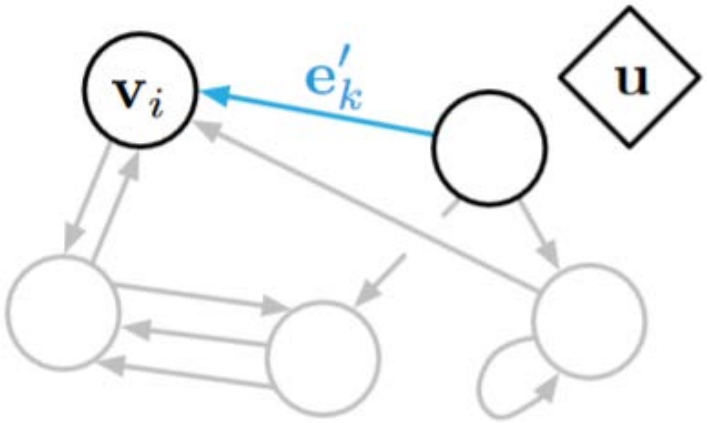

**Global attribute (feature)**


KAIST

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Graph neural network

**GNN blocks**




(a) Edge update

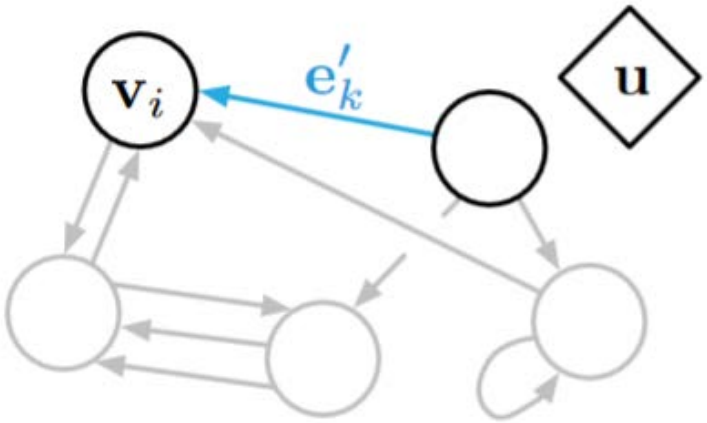
$$\mathbf{e}'_k = \mathrm{NN}\left(\mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{e}_k, \mathbf{u}\right)$$

KAIST

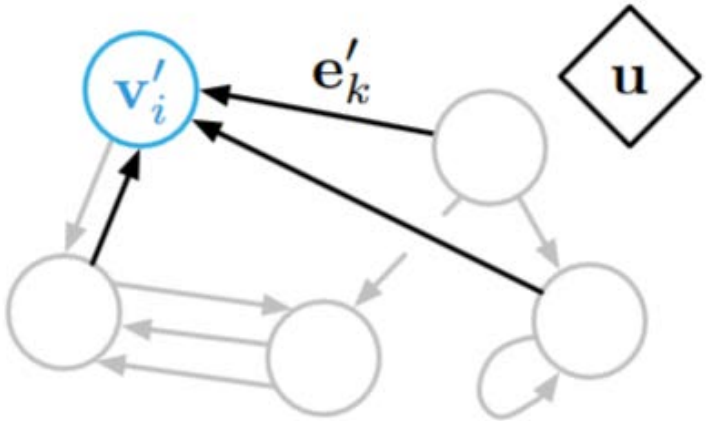
Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Graph neural network

**GNN blocks**




(a) Edge update




(b) Node update

$$\mathbf{e}'_k = \mathrm{NN}\left(\mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{e}_k, \mathbf{u}\right)$$

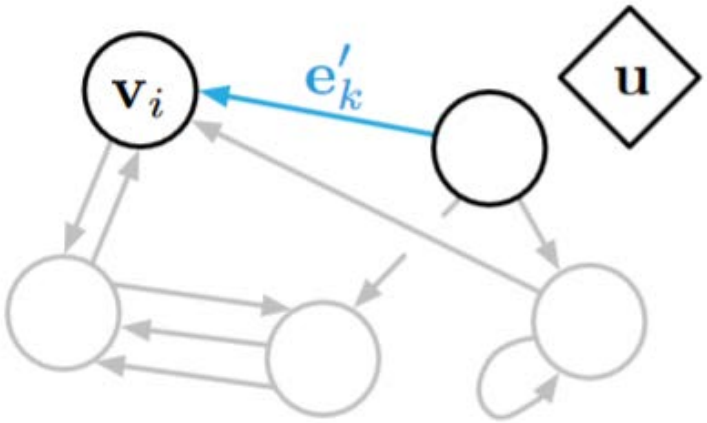
$$\bar{\mathbf{e}}'_i = \sum_{k:r_k=i} \mathbf{e}'_k$$

$$\mathbf{v}'_i = \mathrm{NN}(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$$

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

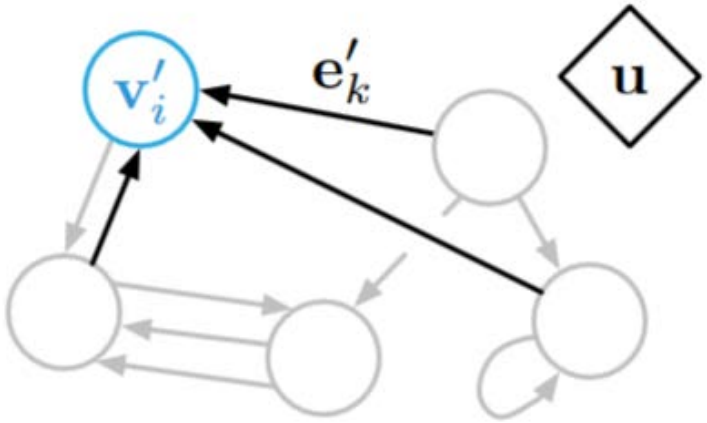
# Graph neural network

**GNN blocks**




(a) Edge update

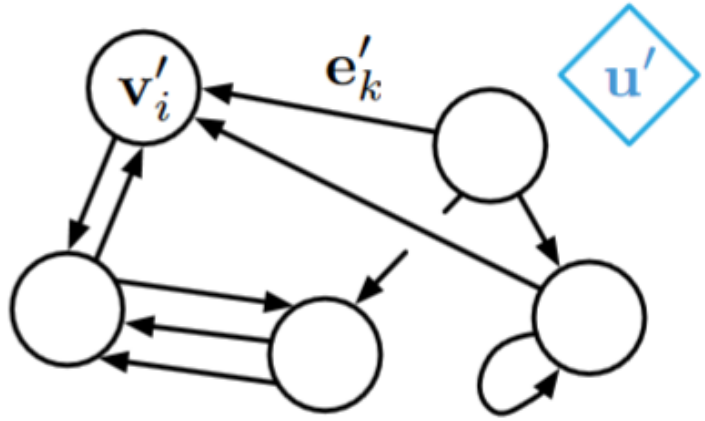
$$\mathbf{e}'_k = \mathrm{NN}\left(\mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{e}_k, \mathbf{u}\right)$$




(b) Node update

$$\bar{\mathbf{e}}'_i = \sum_{k:r_k=i} \mathbf{e}'_k$$

$$\mathbf{v}'_i = \mathrm{NN}(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$$




(c) Global update

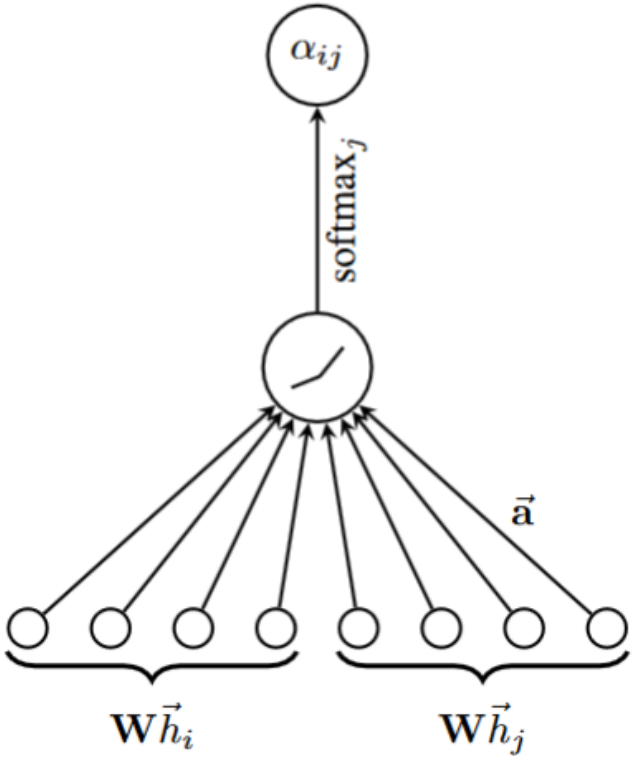
$$\bar{\mathbf{v}}' = \sum_j \mathbf{v}'_j$$

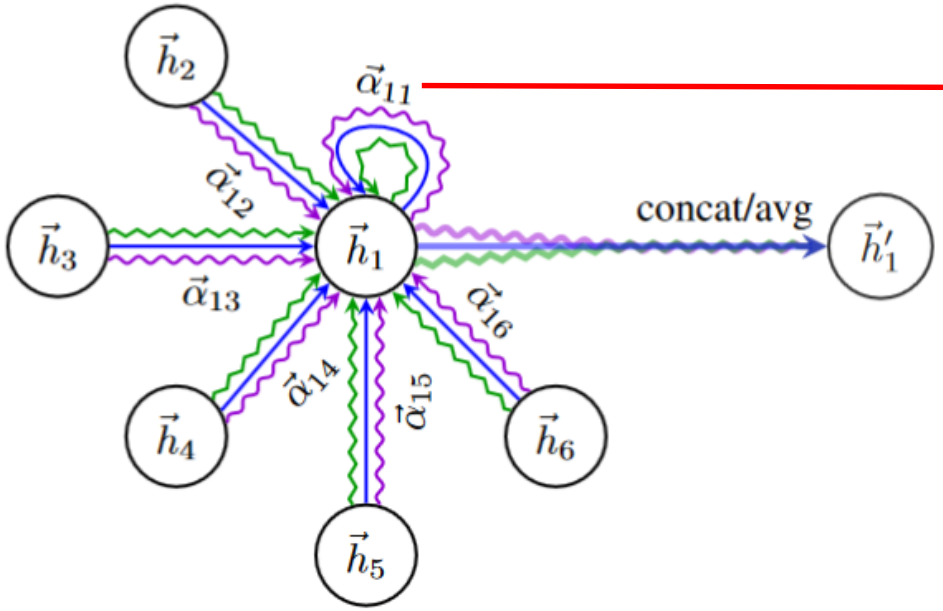
$$\mathbf{u}' = \mathrm{NN}(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$$

KAIST

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

# Graph attention network




$$\mathbf{e}'_k = \mathrm{NN}(\mathbf{v}_{r_k}, \mathbf{v}_{s_k})$$

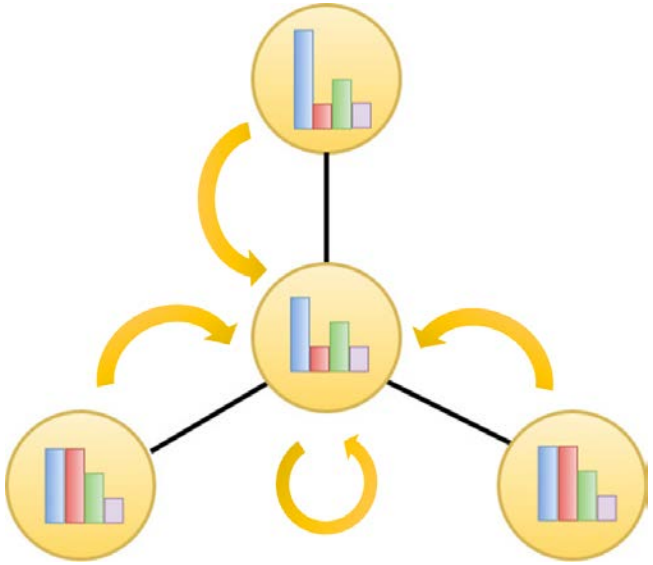
$$\mathbf{v}'_i = \sum_{k:r_k=i} \mathrm{NN}(\mathbf{e}'_k, \mathbf{v}_{s_k})$$



Velickovic, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* 1.2 (2017).

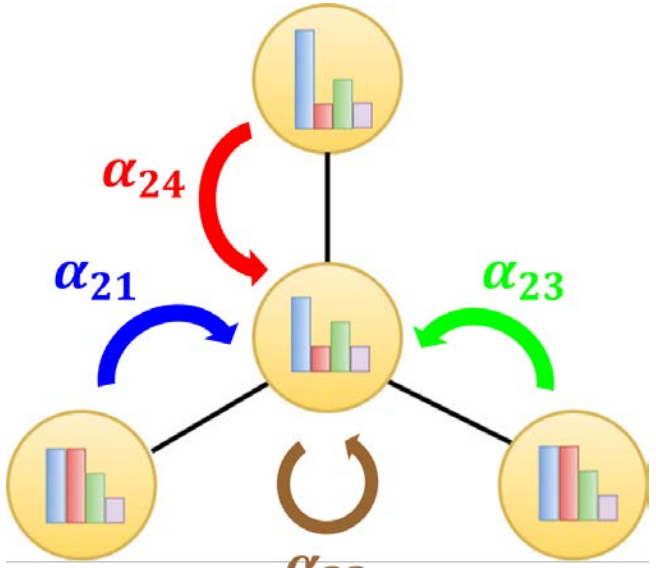
# GCN vs GAT

Vanilla GCN updates information of neighbor atoms **with same importance**.



$$H^{(l+1)} = \sigma\left(\sum_{j \in N(i)} H_j^{(l)} W^{(l)}\right)$$

Attention mechanism enables GCN to update nodes **with different importance.**




$$H^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} H_j^{(l)} W^{(l)}\right)$$

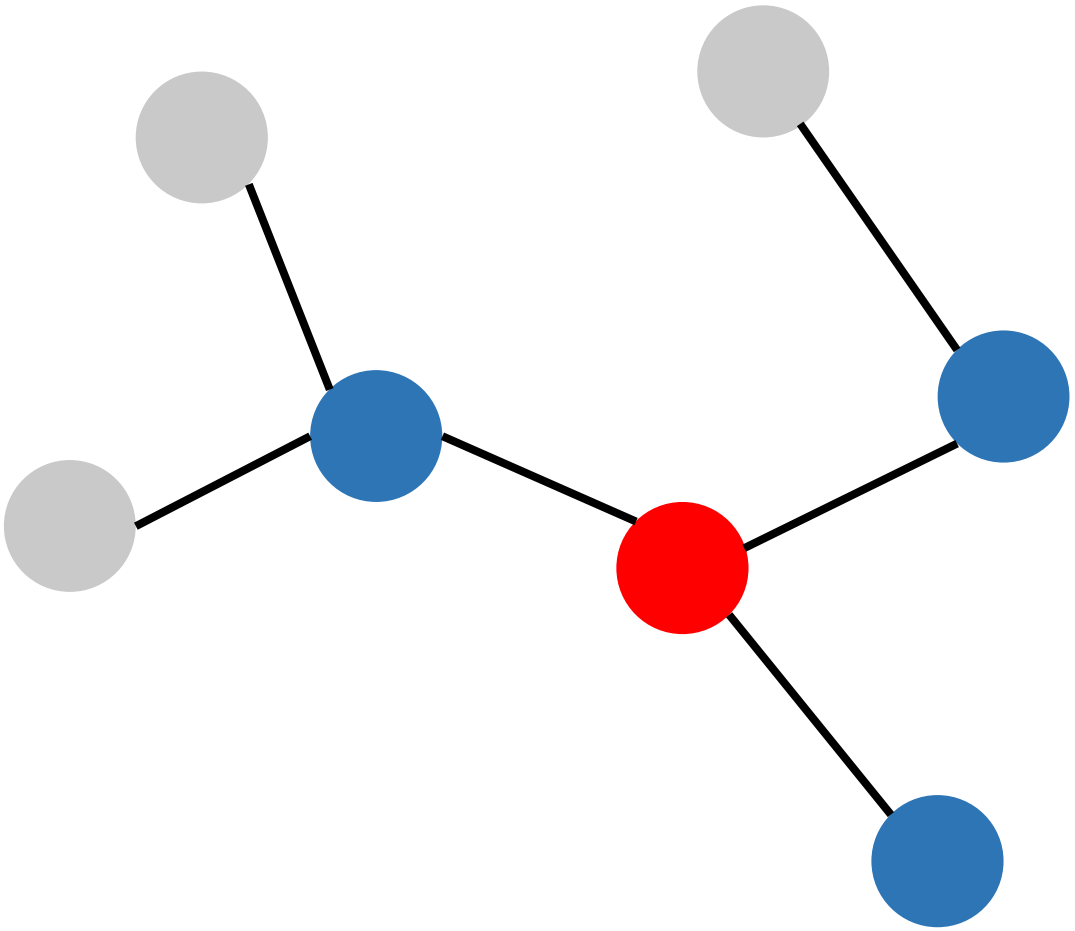
The **attention** is nothing but **edge attributes** which find relations between node attributes



Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261* (2018).

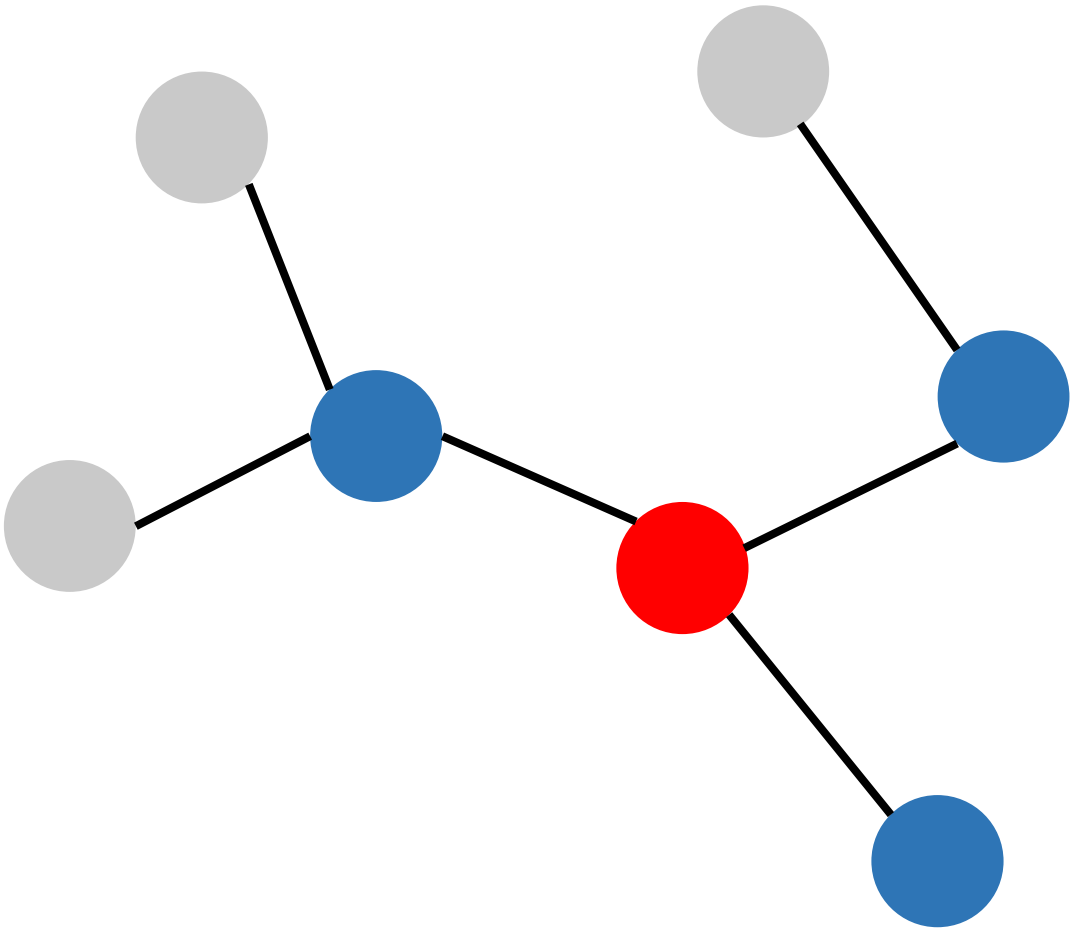
# Message passing neural network

# Message passing neural network



When we update the i-th node state

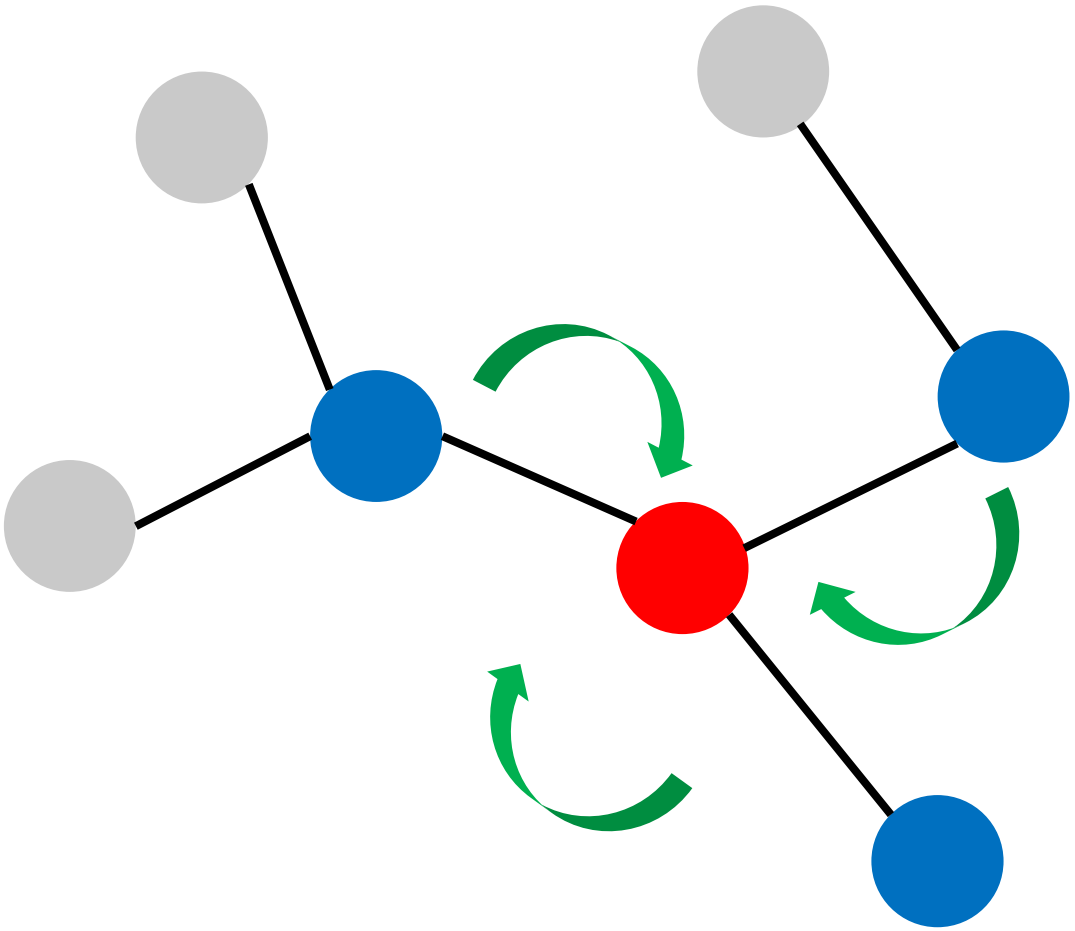
# Message passing neural network



When we update the i-th node state

GCN : $H_i^{(l)} = \sum_{j \in N_i} H_j^{(l)} W^{(l)}$

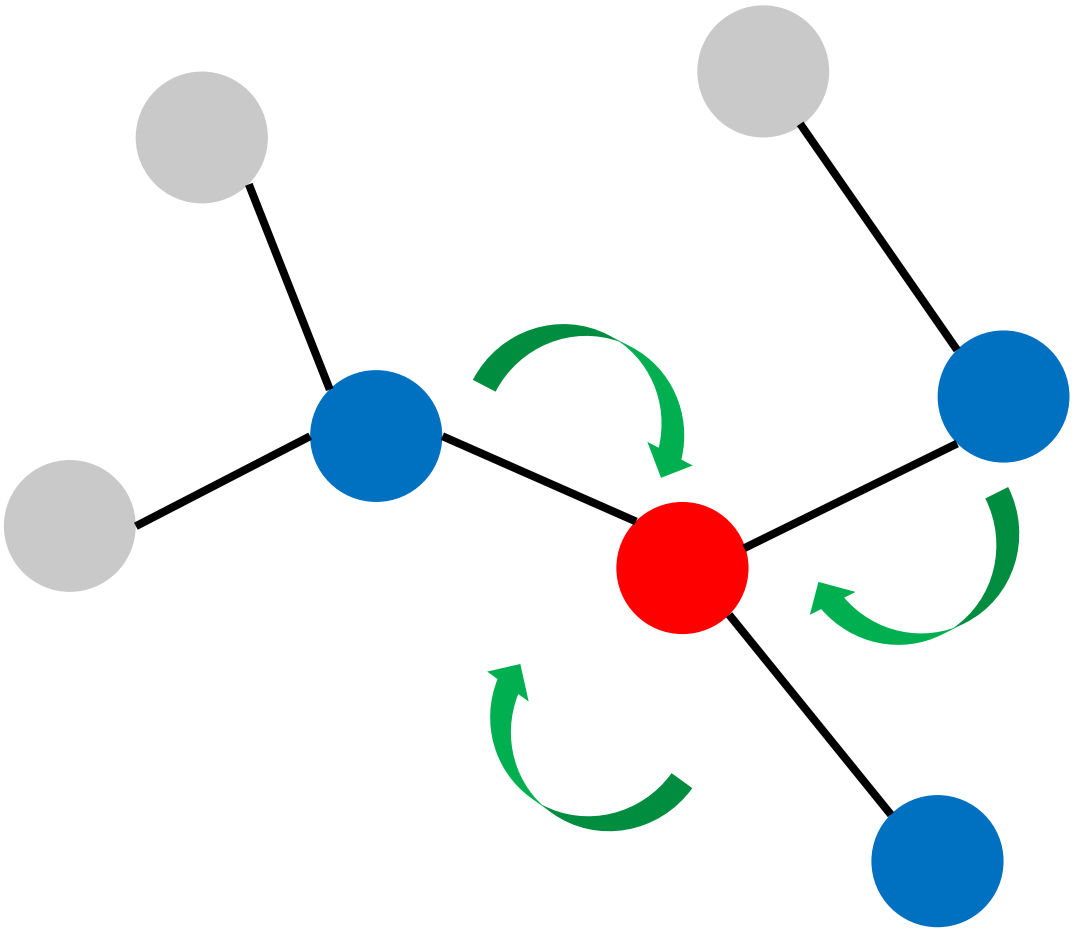
# Message passing neural network



When we update the i-th node state

GCN : $H_i^{(l)} = \sum_{j \in N_i} H_j^{(l)} W^{(l)}$

# Message passing neural network

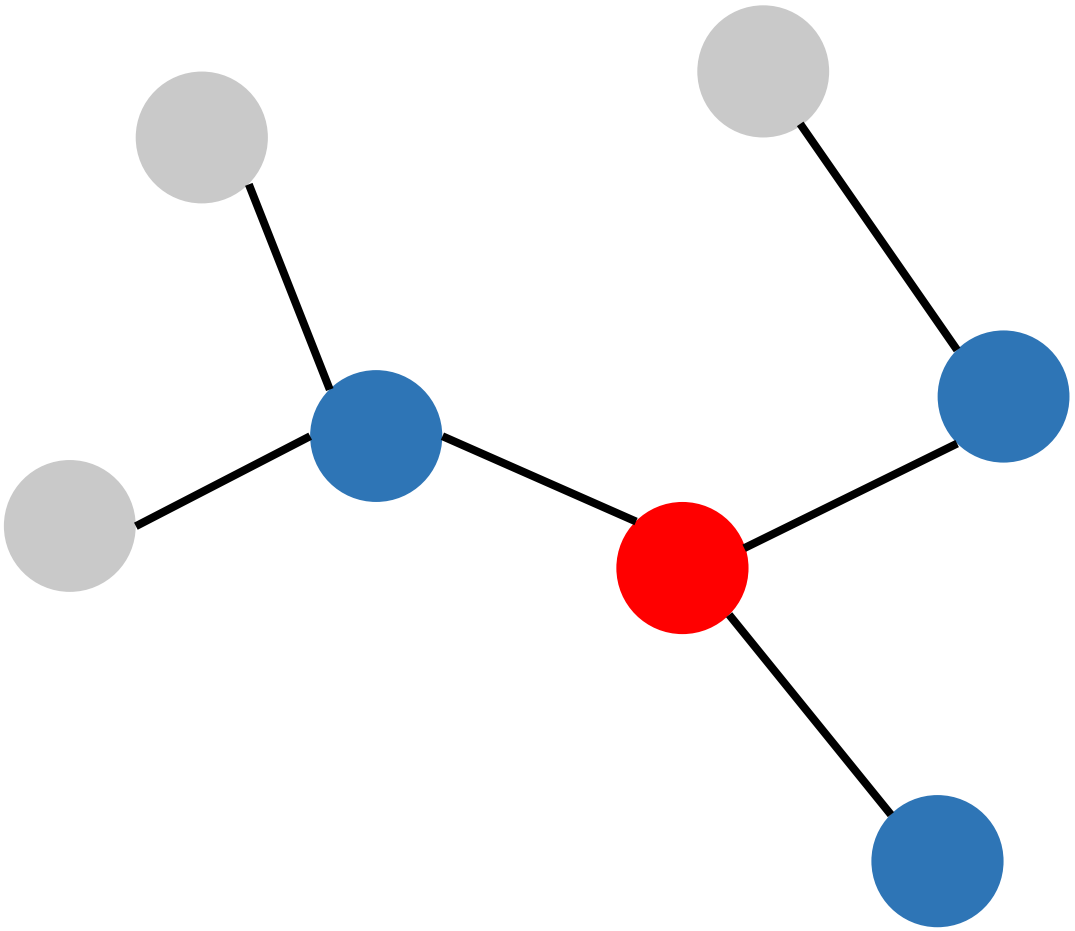


When we update the i-th node state

GCN : $$H_i^{(l)} = \sum_{j \in N_i} H_j^{(l)} W^{(l)}$$

GCN treats the information of adjacent nodes equally.

# Message passing neural network

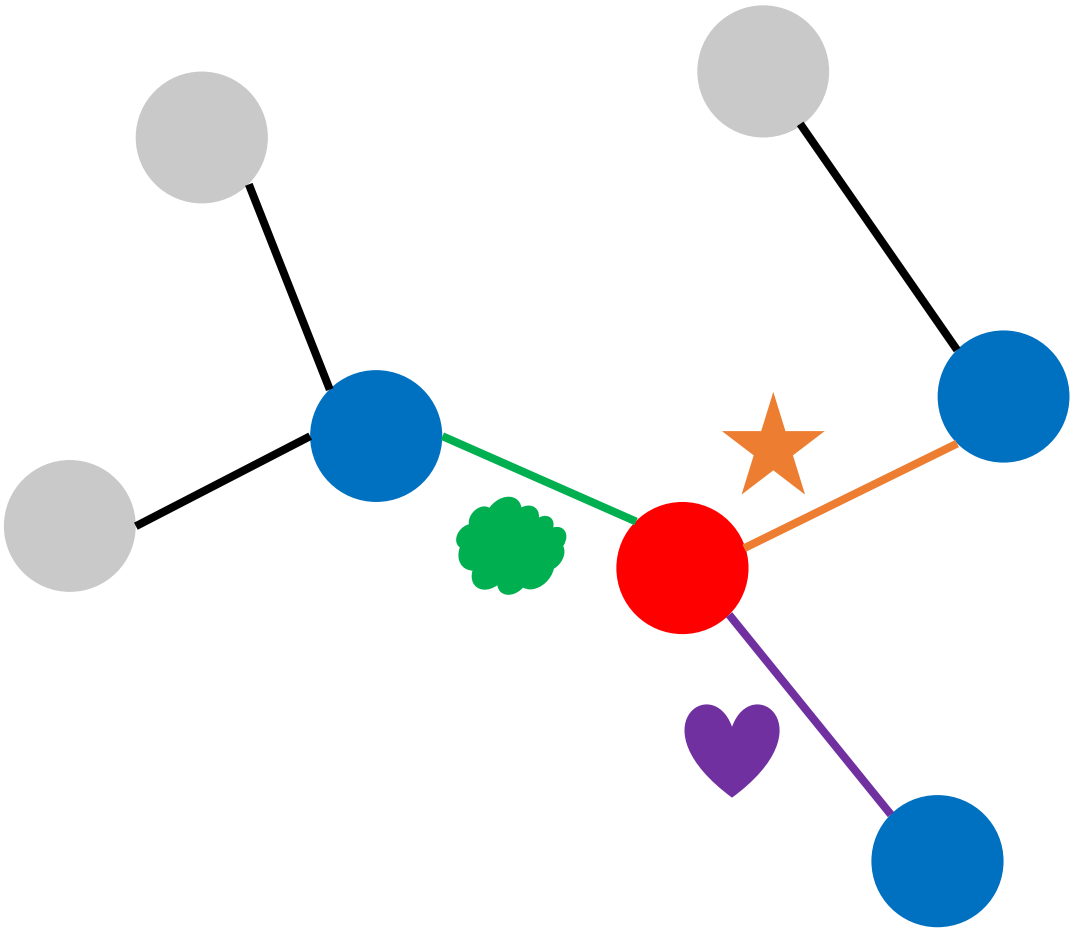


When we update the i-th node state 

MPNN : $H_i^{(l)} = f(H_i^{(l)}, m_i^{(l+1)})$

GCN : $H_i^{(l)} = \sum_{j \in N_i} H_j^{(l)} W^{(l)}$

# Message passing neural network

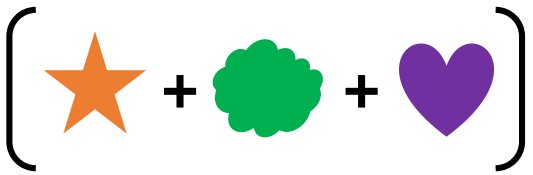


When we update the i-th node state 

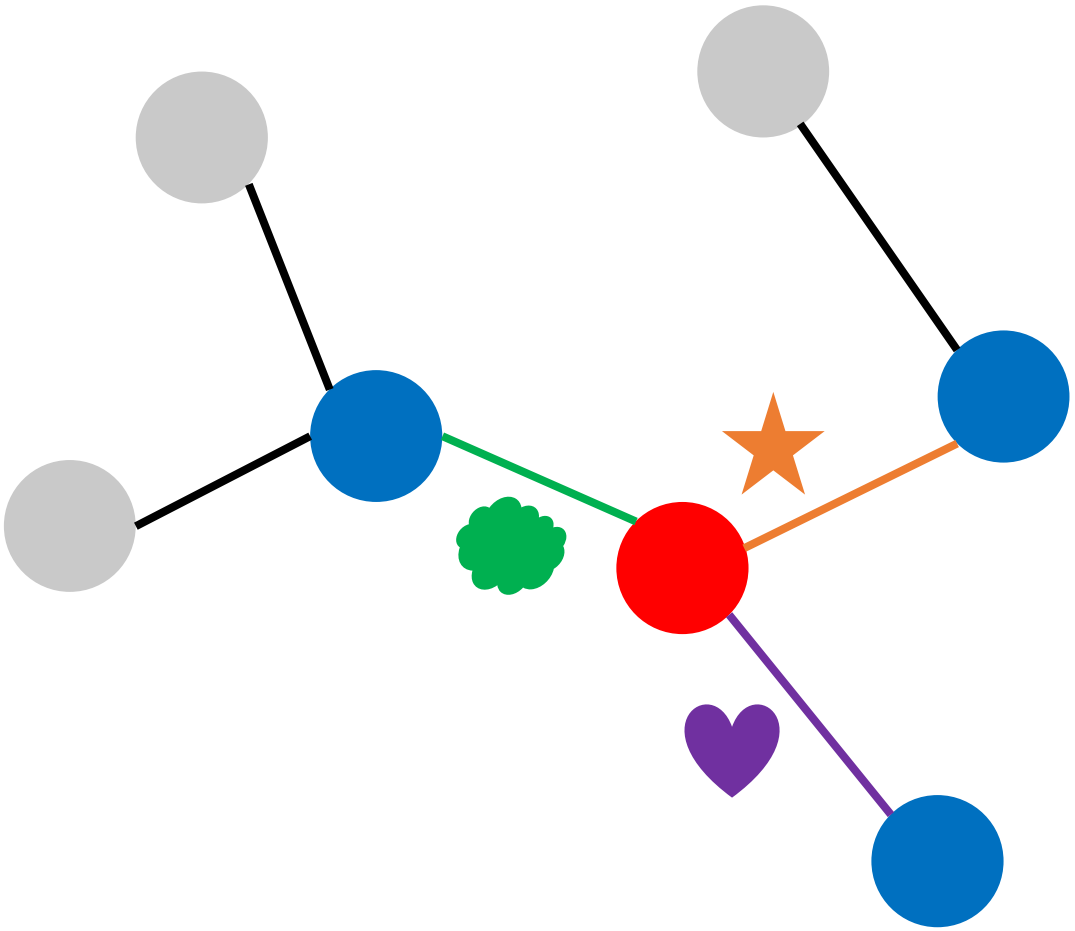
MPNN : $H_i^{(l)} = f(H_i^{(l)}, m_i^{(l+1)})$

$m_{ij}^{(l+1)} =$  $= f($  ,  ,  $)$

The message state $m_{ij}^{(l+1)}$ is updated as a function of the i- and j-th node states and edge features.

$m_i^{(l+1)} =$ 

# Message passing neural network



When we update the i-th node state 

MPNN : $H_i^{(l)} = f(H_i^{(l)}, m_i^{(l+1)})$

$$m_{ij}^{(l+1)} = M^{(l)}\left(H_i^{(l)}, H_j^{(l)}, e_{ij}\right)$$

$e_{ij}$ : ex) single/double/aromatic/… bond

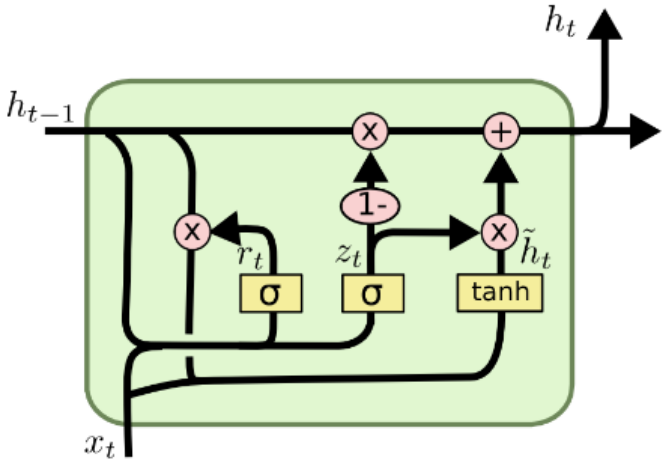
$$m_i^{(l+1)} = \sum_{j \in N_i} m_{ij}^{(l+1)}$$

$$H_i^{(l)} = \mathrm{GRU}(H_i^{(l)}, m_i^{(l+1)})$$

KAIST

# Message passing neural network

When we update the i-th node state




In this case, $x_t = m_i^{(l+1)}$

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

MPNN : $H_i^{(l)} = f(H_i^{(l)}, m_i^{(l+1)})$

$$m_{ij}^{(l+1)} = M^{(l)}\left(H_i^{(l)}, H_j^{(l)}, e_{ij}\right)$$

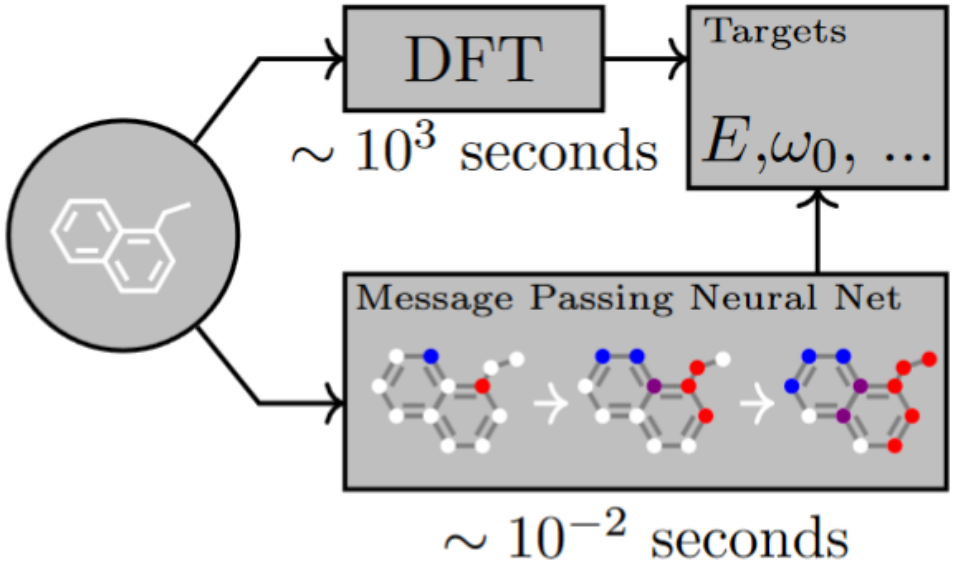
$e_{ij}$ : ex) single/double/aromatic/… bond

$$m_i^{(l+1)} = \sum_{j \in N_i} m_{ij}^{(l+1)}$$

$$H_i^{(l)} = \mathrm{GRU}(H_i^{(l)}, m_i^{(l+1)})$$

# MPNN for quantum chemistry

# MPNN for quantum chemistry




Principle-based approach

- Do not need data, but high-level computational methods are required for accurate calculations.
- Density functional theory (DFT) is commonly used
- $10^3 \sim 10^8$ sec, depend on the number of atoms (electrons)

Statistical approach

- Extremely fast in calculation of molecular properties
- Qualified data (e.g. QM9 dataset) is required

Gilmer, Justin, et al. "Neural message passing for quantum chemistry." *arXiv preprint arXiv:1704.01212* (2017).

# MPNN for quantum chemistry

- We develop an MPNN which achieves state of the art results on all 13 targets and predicts DFT to within chemical accuracy on 11 out of 13 targets.

- We develop several different MPNNs which predict DFT to within chemical accuracy on 5 out of 13 tar- gets while operating on the topology of the molecule alone (with no spatial information as input).

- We develop a general method to train MPNNs with larger node representations without a corresponding increase in computation time or memory, yielding a substantial savings over previous MPNNs for high dimensional node representations.

# MPNN for quantum chemistry

## 2. Message Passing Neural Networks

There are at least eight notable examples of models from the literature that we can describe using our Message Passing Neural Networks (MPNN) framework. For simplicity we describe MPNNs which operate on undirected graphs $G$ with node features $x_v$ and edge features $e_{vw}$. It is trivial to extend the formalism to directed multigraphs. The forward pass has two phases, a message passing phase and a readout phase. The message passing phase runs for $T$ time steps and is defined in terms of message functions $M_t$ and vertex update functions $U_t$. During the message passing phase, hidden states $h_v^t$ at each node in the graph are updated based on messages $m_v^{t+1}$ according to

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{1}$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{2}$$

where in the sum, $N(v)$ denotes the neighbors of $v$ in graph $G$. The readout phase computes a feature vector for the whole graph using some readout function $R$ according to

$$\hat{y} = R(\{h_v^T \mid v \in G\}). \tag{3}$$

The message functions $M_t$, vertex update functions $U_t$, and readout function $R$ are all learned differentiable functions.

# MPNN for quantum chemistry

**Convolutional Networks for Learning Molecular Fingerprints, Duvenaud et al. (2015)**

The message function used is $M(h_v, h_w, e_{vw}) = (h_w, e_{vw})$ where $(.,.)$ denotes concatenation. The vertex update function used is $U_t(h_v^t, m_v^{t+1}) = \sigma(H_t^{\deg(v)} m_v^{t+1})$, where $\sigma$ is the sigmoid function, $\deg(v)$ is the degree of vertex $v$ and $H_t^N$ is a learned matrix for each time step $t$ and vertex degree $N$. $R$ has skip connections to all previous hidden states $h_v^t$ and is equal to $f\left(\sum_{v,t} \text{softmax}(W_t h_v^t)\right)$, where $f$ is a neural network and $W_t$ are learned readout matrices, one for each time step $t$. This message passing scheme may be problematic since the resulting message vector is $m_v^{t+1} = (\sum h_w^t, \sum e_{vw})$, which separately sums over connected nodes and connected edges. It follows that the message passing implemented in Duvenaud et al. (2015) is unable to identify correlations between edge states and node states.

**Gated Graph Neural Networks (GG-NN), Li et al. (2016)**

The message function used is $M_t(h_v^t, h_w^t, e_{vw}) = A_{e_{vw}} h_w^t$, where $A_{e_{vw}}$ is a learned matrix, one for each edge label $e$ (the model assumes discrete edge types). The update function is $U_t = \text{GRU}(h_v^t, m_v^{t+1})$, where GRU is the Gated

Recurrent Unit introduced in Cho et al. (2014). This work used weight tying, so the same update function is used at each time step $t$. Finally,

$$R = \sum_{v \in V} \sigma\left(i(h_v^{(T)}, h_v^0)\right) \odot \left(j(h_v^{(T)})\right) \tag{4}$$

where $i$ and $j$ are neural networks, and $\odot$ denotes element-wise multiplication.

# MPNN for quantum chemistry

*Table 1.* Atom Features

| Feature | Description |
|---|---|
| Atom type | H, C, N, O, F (one-hot) |
| Atomic number | Number of protons (integer) |
| Acceptor | Accepts electrons (binary) |
| Donor | Donates electrons (binary) |
| Aromatic | In an aromatic system (binary) |
| Hybridization | sp, sp2, sp3 (one-hot or null) |
| Number of Hydrogens | (integer) |

# MPNN for quantum chemistry

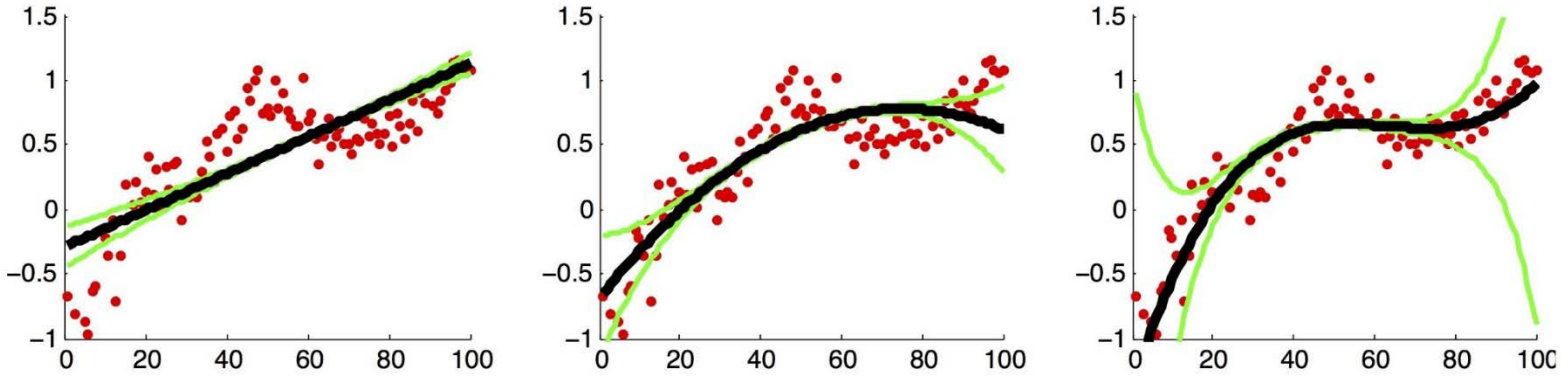
*Table 2.* Comparison of Previous Approaches (left) with MPNN baselines (middle) and our methods (right)

| Target | BAML | BOB | CM | ECFP4 | HDAD | GC | GG-NN | DTNN | enn-s2s | enn-s2s-ens5 |
|---|---|---|---|---|---|---|---|---|---|---|
| mu | 4.34 | 4.23 | 4.49 | 4.82 | 3.34 | 0.70 | 1.22 | - | **0.30** | 0.20 |
| alpha | 3.01 | 2.98 | 4.33 | 34.54 | 1.75 | 2.27 | 1.55 | - | **0.92** | 0.68 |
| HOMO | 2.20 | 2.20 | 3.09 | 2.89 | 1.54 | 1.18 | 1.17 | - | **0.99** | 0.74 |
| LUMO | 2.76 | 2.74 | 4.26 | 3.10 | 1.96 | 1.10 | 1.08 | - | **0.87** | 0.65 |
| gap | 3.28 | 3.41 | 5.32 | 3.86 | 2.49 | 1.78 | 1.70 | - | **1.60** | 1.23 |
| R2 | 3.25 | 0.80 | 2.83 | 90.68 | 1.35 | 4.73 | 3.99 | - | **0.15** | 0.14 |
| ZPVE | 3.31 | 3.40 | 4.80 | 241.58 | 1.91 | 9.75 | 2.52 | - | **1.27** | 1.10 |
| U0 | 1.21 | 1.43 | 2.98 | 85.01 | 0.58 | 3.02 | 0.83 | - | **0.45** | 0.33 |
| U | 1.22 | 1.44 | 2.99 | 85.59 | 0.59 | 3.16 | 0.86 | - | **0.45** | 0.34 |
| H | 1.22 | 1.44 | 2.99 | 86.21 | 0.59 | 3.19 | 0.81 | - | **0.39** | 0.30 |
| G | 1.20 | 1.42 | 2.97 | 78.36 | 0.59 | 2.95 | 0.78 | .84[2] | **0.44** | 0.34 |
| Cv | 1.64 | 1.83 | 2.36 | 30.29 | 0.88 | 1.45 | 1.19 | - | **0.80** | 0.62 |
| Omega | 0.27 | 0.35 | 1.32 | 1.47 | 0.34 | 0.32 | 0.53 | - | **0.19** | 0.15 |
| Average | 2.17 | 2.08 | 3.37 | 53.97 | 1.35 | 2.59 | 1.36 | - | **0.68** | 0.52 |

$R^2$: the electronic spatial extent (Bohr$^2$),

# Summary

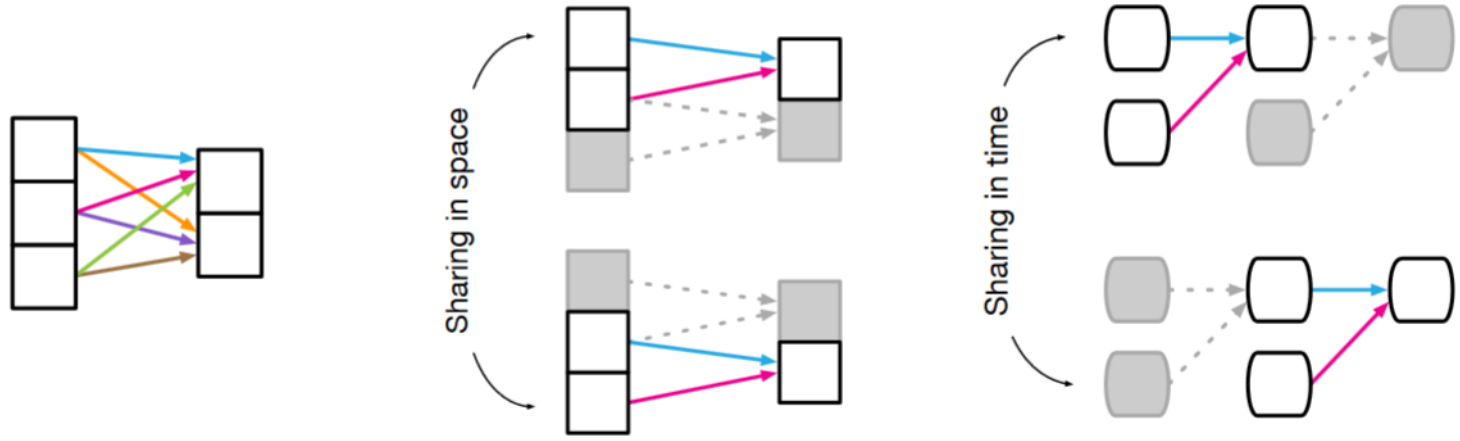
- Relational reasoning, then, involves manipulating **structured representations** of **entities** and **relations**, using **rules** for how they can be composed. We use these terms to capture notions from cognitive science, theoretical computer science, and AI
- We explored how using **relational inductive biases** within deep learning architectures can facilitate learning about entities, relations, and rules for composing them
- An inductive bias allows a learning algorithm to prioritize one solution (or interpretation) over another, independent of the observed data.




**Bias-variance tradeoff**

# Summary

**Inductive bias is another name of weight sharing**

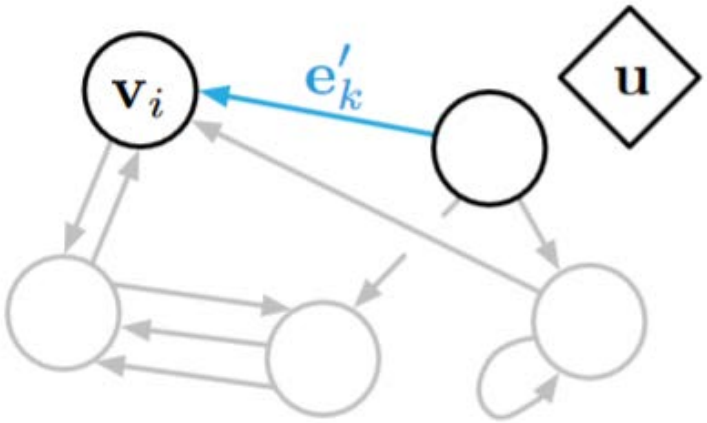



(a) Fully connected (b) Convolutional (c) Recurrent

| Component | Entities | Relations | Rel. inductive bias | Invariance |
|---|---|---|---|---|
| Fully connected | Units | All-to-all | Weak | - |
| Convolutional | Grid elements | Local | Locality | Spatial translation |
| Recurrent | Timesteps | Sequential | Sequentiality | Time translation |
| Graph network | Nodes | Edges | Arbitrary | Node, edge permutations |

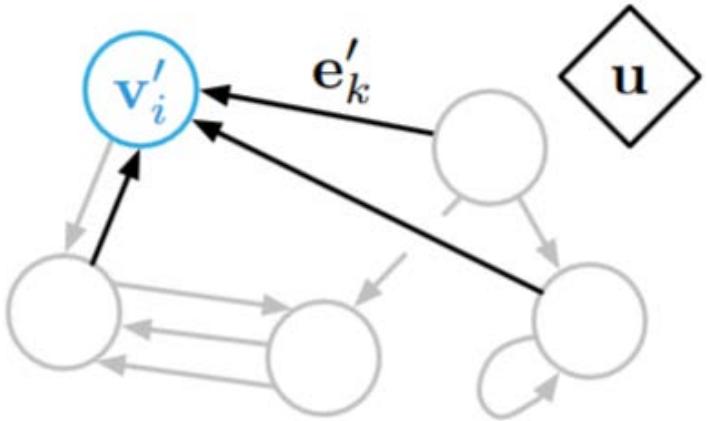
# Summary

**GNN blocks**




(a) Edge update

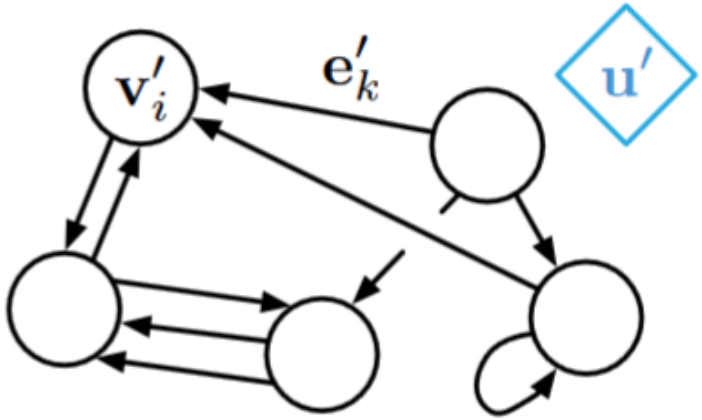
$$\mathbf{e}'_k = \mathrm{NN}\left(\mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{e}_k, \mathbf{u}\right)$$




(b) Node update

$$\bar{\mathbf{e}}'_i = \sum_{k:r_k=i} \mathbf{e}'_k$$

$$\mathbf{v}'_i = \mathrm{NN}(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$$




(c) Global update

$$\bar{\mathbf{v}}' = \sum_j \mathbf{v}'_j$$

$$\mathbf{u}' = \mathrm{NN}(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$$

# New terms

- Structure, Entity, Relation, Rule
- Inductive biases
- Weak or strong prior
- Node attribute(feature)
- Edge attribute(feature)
- Global attribute(feature)
- Message passing neural network