University of Amsterdam

AMLAB Amsterdam Machine Learning Lab

Group Equivariant Deep Learning

Lecture 3 - Equivariant graph neural networks

Lecture 3.3 - Tensor products as conditional linear layers (and MLPs)

A motivation for attributed conditioned message passing using bilinear layers

Erik Bekkers, Amsterdam Machine Learning Lab, University of Amsterdam

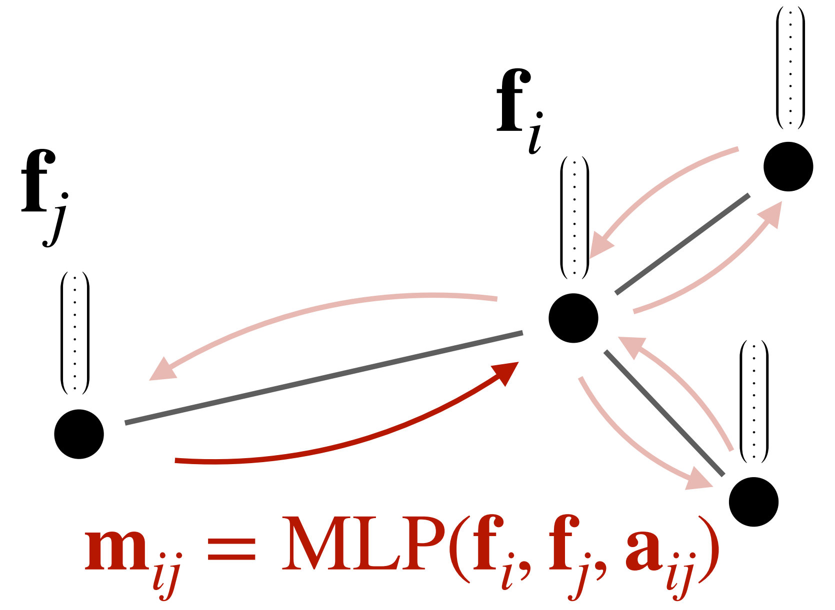
This mini-course serves as a module with the UvA Master AI course Deep Learning 2 https://uvadl2c.github.io/

Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$\forall_{\mathbf{R}\in SO(3)}:$

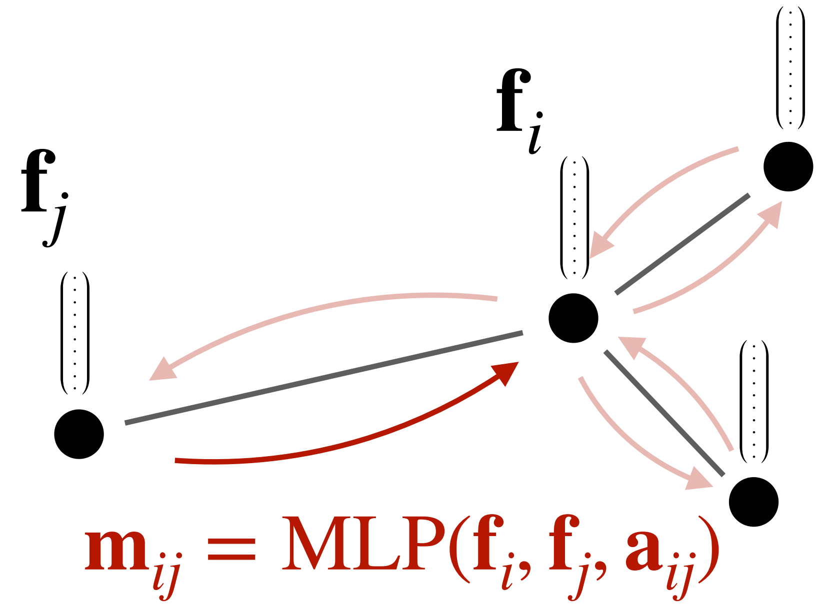
$$\phi(\rho^{in}(\mathbf{R})\mathbf{v}) = \rho^{out}(\mathbf{R})\phi(\mathbf{v})$$



Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$$\forall_{\mathbf{R}\in SO(3)}: \quad \phi(\rho^{in}(\mathbf{R})\mathbf{v}) = \rho^{out}(\mathbf{R})\phi(\mathbf{v})$$

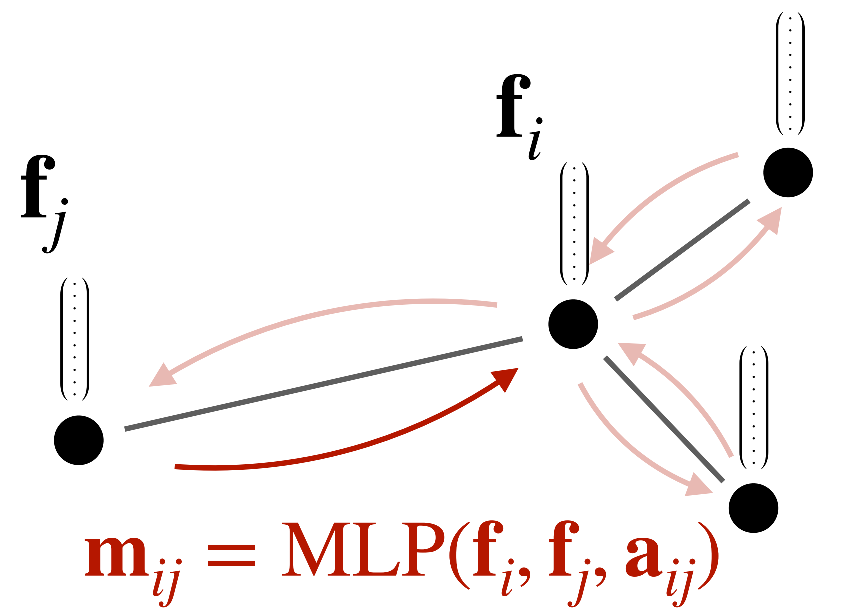


Problem: We like to parametrize $\phi(\mathbf{v}) = \mathrm{MLP}(\mathbf{v})$ with multi-layer perceptrons, however, they can only handle scalar-valued vectors

Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$$\forall_{\mathbf{R}\in SO(3)}: \quad \mathrm{MLP}(\rho^{in}(\mathbf{R})\mathbf{v}) = \rho^{out}(\mathbf{R})\mathrm{MLP}(\mathbf{v})$$

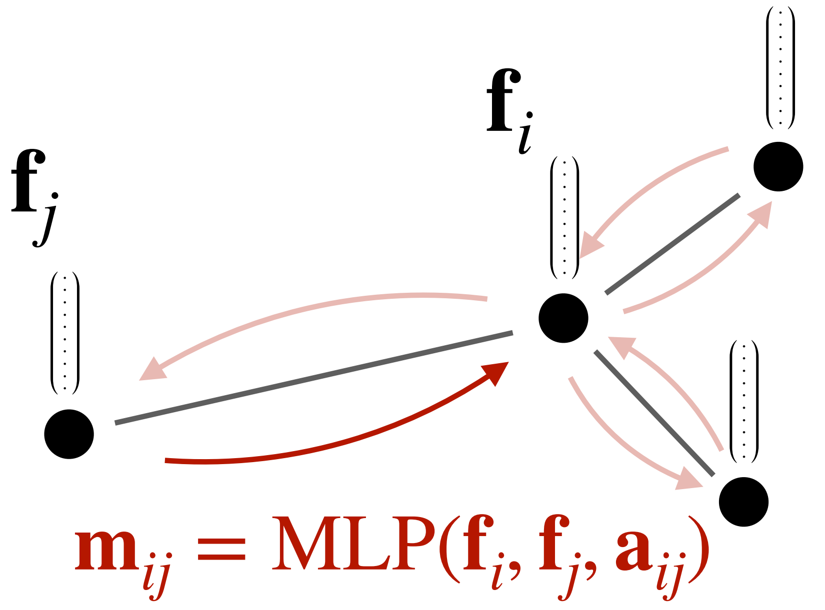


Problem: We like to parametrize $\phi(\mathbf{v}) = \mathrm{MLP}(\mathbf{v})$ with multi-layer perceptrons,
however, they can only handle scalar-valued vectors

Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$$\forall_{\mathbf{R}\in SO(3)}: \quad \text{MLP}(\rho^{in}(\mathbf{R})\mathbf{v}) = \rho^{out}(\mathbf{R})\text{MLP}(\mathbf{v})$$



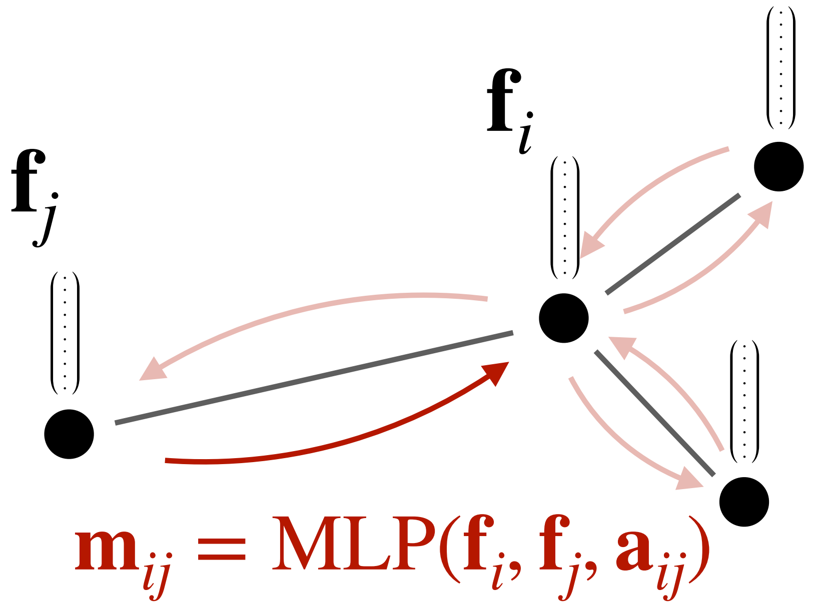
Problem: We like to parametrize $\phi(\mathbf{v}) = \text{MLP}(\mathbf{v})$ with multi-layer perceptrons, however, they can only handle scalar-valued vectors

- Scalars $v \in \mathbb{R}$ trivially transform, i.e., $\rho_0(\mathbf{R})v = 1 \cdot v = v$
- Thus any vector $\mathbf{v} \in \mathbb{R}^C$ of scalars transforms via $\rho(\mathbf{R}) = [\bigoplus^C \rho_0](\mathbf{R}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$\forall_{\mathbf{R}\in SO(3)}$: $$\text{MLP}(\rho^{in}(\mathbf{R})\mathbf{v}) = \rho^{out}(\mathbf{R})\text{MLP}(\mathbf{v})$$



Problem: We like to parametrize $\phi(\mathbf{v}) = \text{MLP}(\mathbf{v})$ with multi-layer perceptrons, however, they can only handle scalar-valued vectors

- Scalars $v \in \mathbb{R}$ trivially transform, i.e., $\rho_0(\mathbf{R})v = 1 \cdot v = v$
- Thus any vector $\mathbf{v} \in \mathbb{R}^C$ of scalars transforms via $\rho(\mathbf{R}) = [\bigoplus^C \rho_0](\mathbf{R}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

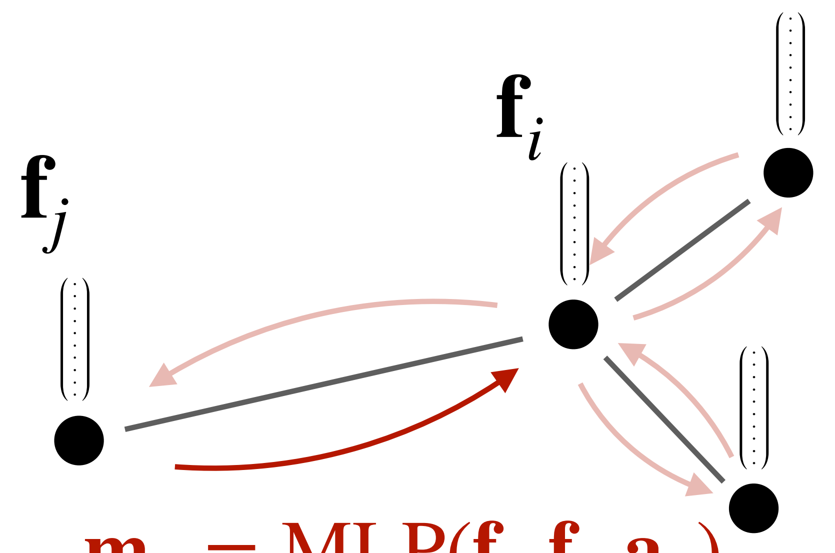
So whatever the MLP takes as input, it should be invariant to rotations, i.e.,

$$\rho^{in} = \rho^{out} = \text{Id}$$

Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$$\forall_{\mathbf{R}\in SO(3)}: \quad \mathrm{MLP}(\rho^{in}(\mathbf{R})\mathbf{v}\,|\,\mathbf{a}) = \rho^{out}(\mathbf{R})\mathrm{MLP}(\mathbf{v}\,|\,\mathbf{a})$$



Problem: We like to parametrize $\phi(\mathbf{v}) = \mathrm{MLP}(\mathbf{v})$ with multi-layer perceptrons, however, they can only handle scalar-valued vectors

- Scalars $v \in \mathbb{R}$ trivially transform, i.e., $\rho_0(\mathbf{R})v = 1 \cdot v = v$
- Thus any vector $\mathbf{v} \in \mathbb{R}^C$ of scalars transforms via $\rho(\mathbf{R}) = [\oplus^C \rho_0](\mathbf{R}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

So whatever the MLP takes as input, it should be invariant to rotations, i.e.,

$$\rho^{in} = \rho^{out} = \mathrm{Id}$$

Thus also any pair-wise attribute embeddings $\mathbf{a}_{ij} = a(\mathbf{x}_j, \mathbf{x}_j) = Y(\mathbf{x}_j - \mathbf{x}_i)$ should be invariant

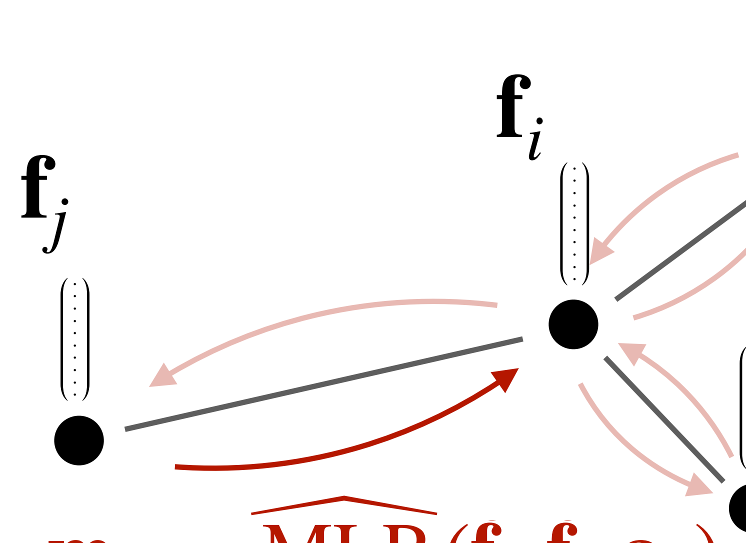
$$\forall_{\mathbf{R}\in SO(3)}: \quad Y(\mathbf{x}_j - \mathbf{x}_i) = Y(\mathbf{R}(\mathbf{x}_j - \mathbf{x}_i))$$

E.g., $Y(\mathbf{x}_j - \mathbf{x}_i) = \|\mathbf{x}_j - \mathbf{x}_i\|$

Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$$\forall_{\mathbf{R}\in SO(3)}: \quad \widehat{\mathrm{MLP}}\,(\rho^{in}(\mathbf{R})\mathbf{v}\,|\,\mathbf{a}) = \rho^{out}(\mathbf{R})\,\widehat{\mathrm{MLP}}\,(\mathbf{v}\,|\,\mathbf{a})$$



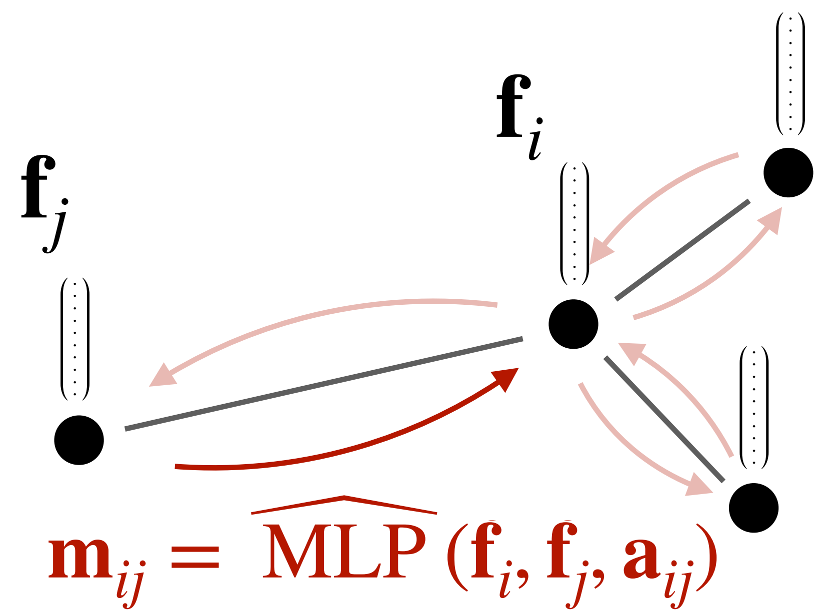
Problem: We like to parametrize $\phi(\mathbf{x}) = \mathrm{MLP}(\mathbf{x})$ with multi-layer perceptrons, however, they can only handle scalar-valued vectors

Solution: Use *equivariant multi-layer perceptrons* $\widehat{\mathrm{MLP}}$

Objective: $SO(3)$ equivariant MLPs

We are looking for (update/message) functions that are equivariant to $SO(3)$ transformations

$$\forall_{\mathbf{R}\in SO(3)}: \quad \widehat{\mathrm{MLP}}\,(\rho^{in}(\mathbf{R})\mathbf{v}\,|\,\mathbf{a}) = \rho^{out}(\mathbf{R})\,\widehat{\mathrm{MLP}}\,(\mathbf{v}\,|\,\mathbf{a})$$



Problem: We like to parametrize $\phi(\mathbf{x}) = \mathrm{MLP}(\mathbf{x})$ with multi-layer perceptrons, however, they can only handle scalar-valued vectors

Solution: Use *equivariant multi-layer perceptrons* $\widehat{\mathrm{MLP}}$

Pair-wise attribute embeddings $\mathbf{a}_{ij} = a(\mathbf{x}_j, \mathbf{x}_j) = Y(\mathbf{x}_j - \mathbf{x}_i)$ can now be equivariant

$$\forall_{\mathbf{R}\in SO(3)}: \quad \rho^{in}(\mathbf{R})\,Y(\mathbf{x}_j - \mathbf{x}_i) = Y(\mathbf{R}(\mathbf{x}_j - \mathbf{x}_i))$$

So an $SO(3)$ steerable function!

How to condition MLPs?

Conditional MLP

$$\mathrm{MLP}(\mathbf{v} \,|\, \mathbf{a}_{ij})$$

Conditional linear layers:

Stacking (concat) features to the input (as e.g. in EGNN)

$$\mathbf{W} \begin{pmatrix} \mathbf{v} \\ \mathbf{a}_{ij} \end{pmatrix}$$

Adaptive/conditional weights through basis functions (as e.g. in steerable G-CNNs)

$$\mathbf{W}(\mathbf{a}_{ij})\ \mathbf{v}$$

How to condition MLPs?

Conditional MLP

$$\mathrm{MLP}(\mathbf{v} \,|\, \mathbf{a}_{ij})$$

Conditional linear layers:

Stacking (concat) features to the input (as e.g. in EGNN)

$$\mathbf{W} \begin{pmatrix} \mathbf{v} \\ \mathbf{a}_{ij} \end{pmatrix}$$

Adaptive/conditional weights through basis functions (as e.g. in steerable G-CNNs)

$$\mathbf{W}(\mathbf{a}_{ij}) \; \mathbf{v} \qquad \Longleftrightarrow \qquad Y(\mathbf{a}_{ij}) \overset{bilinear}{\mathbf{W}} \; \mathbf{v}$$

How to condition MLPs?

Conditional MLP

$$\mathrm{MLP}(\mathbf{v} \,|\, \mathbf{a}_{ij})$$

Conditional linear layers:

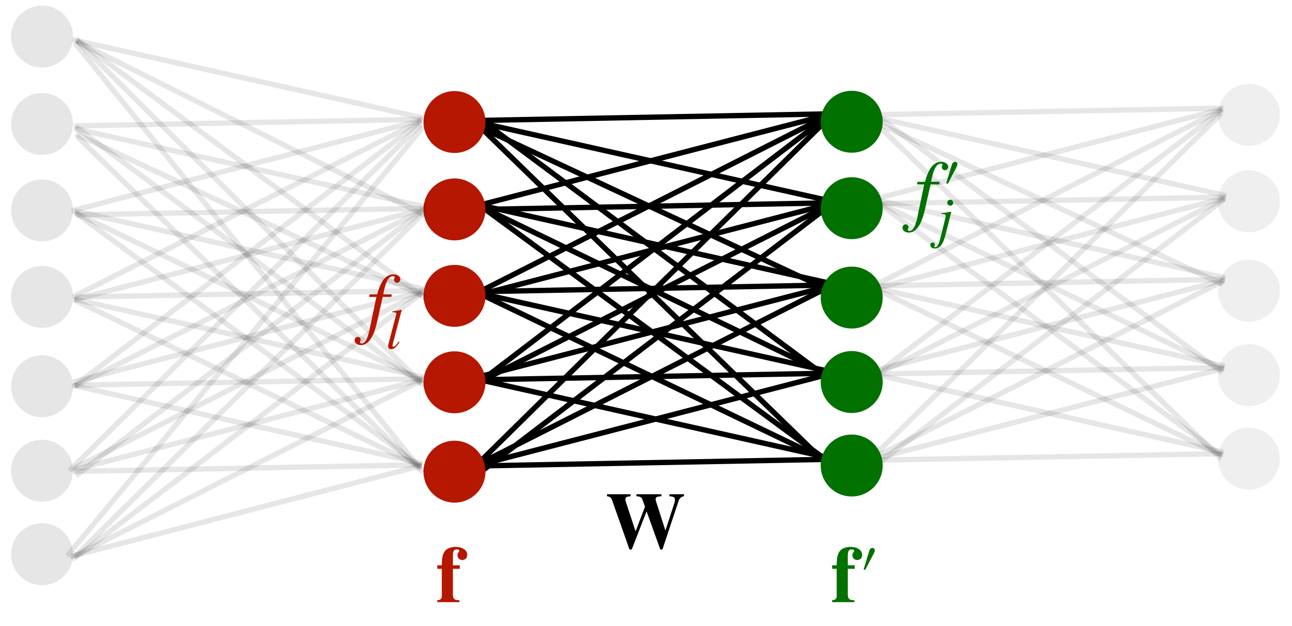
Stacking (concat) features to the input (as e.g. in EGNN)

$$\mathbf{W} \begin{pmatrix} \mathbf{v} \\ \mathbf{a}_{ij} \end{pmatrix}$$

Adaptive/conditional weights through basis functions (as e.g. in steerable G-CNNs)

$$\mathbf{W}(\mathbf{a}_{ij})\ \mathbf{v} \qquad \Longleftrightarrow \qquad Y(\mathbf{a}_{ij})\ \otimes^{\mathbf{W}} \mathbf{v}$$

Conditional linear layers



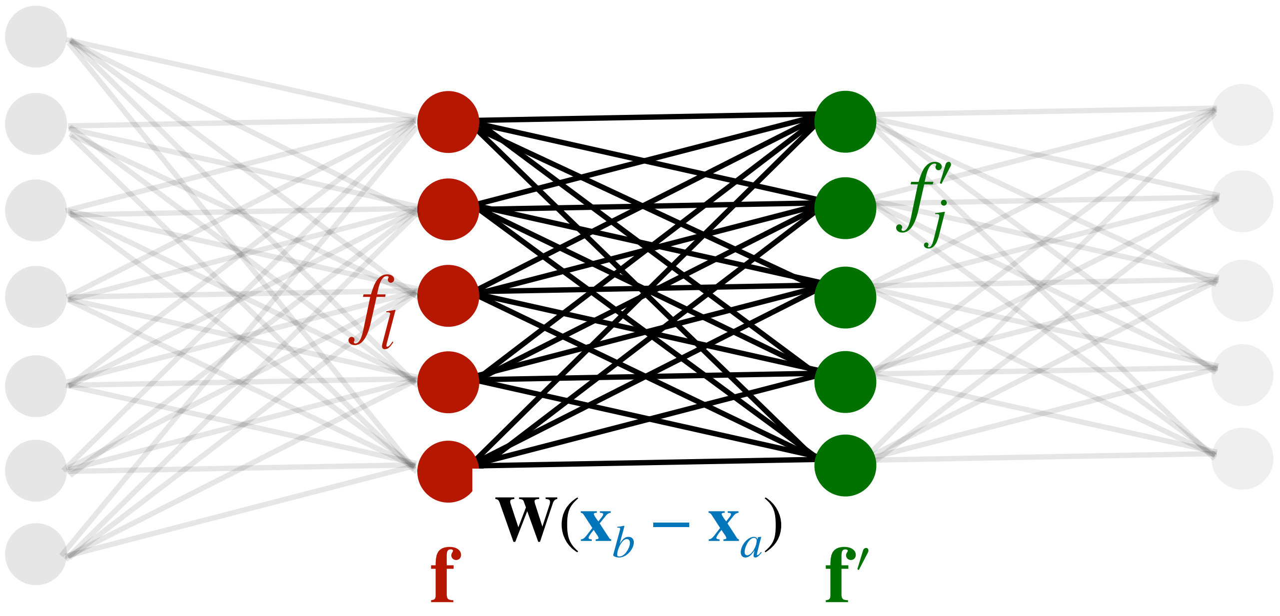
$\mathbf{f} \quad \mapsto \quad \mathbf{f}' = \mathbf{W}\,\mathbf{f} \quad \mapsto \quad \mathbf{f}'' = \sigma(\,\mathbf{f}'\,)$ (Repeat L times)

linear layer   activation

Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\,\mathbf{f} \qquad f'_j = \sum_l w_l^j\, f_l$$

Conditional linear layers



$\mathbf{f} \mapsto \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \mapsto \mathbf{f}'' = \sigma(\,\mathbf{f}'\,)$ (Repeat L times)

linear layer — activation

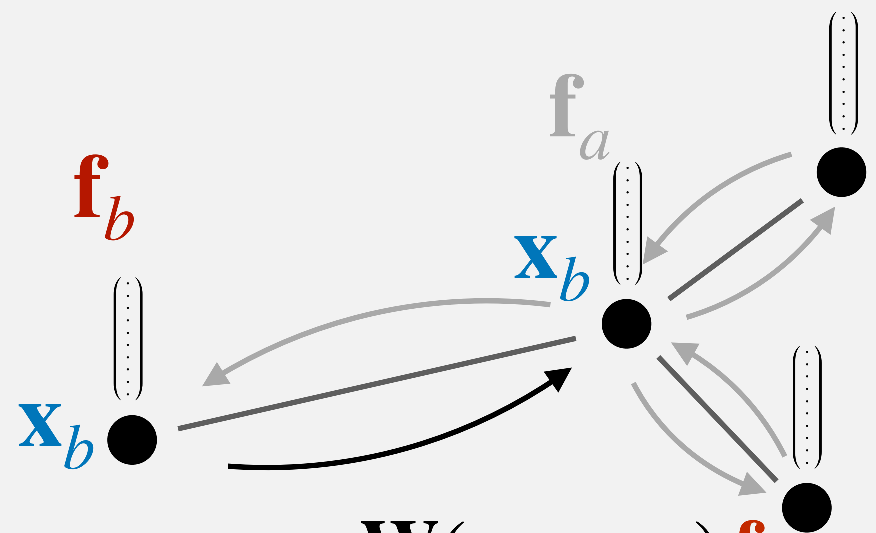
Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\ \mathbf{f} \qquad f'_j = \sum_l w^j_l\, f_l$$

Conditional linear layer (weight matrix depends on $\mathbf{x}_b - \mathbf{x}_a$)

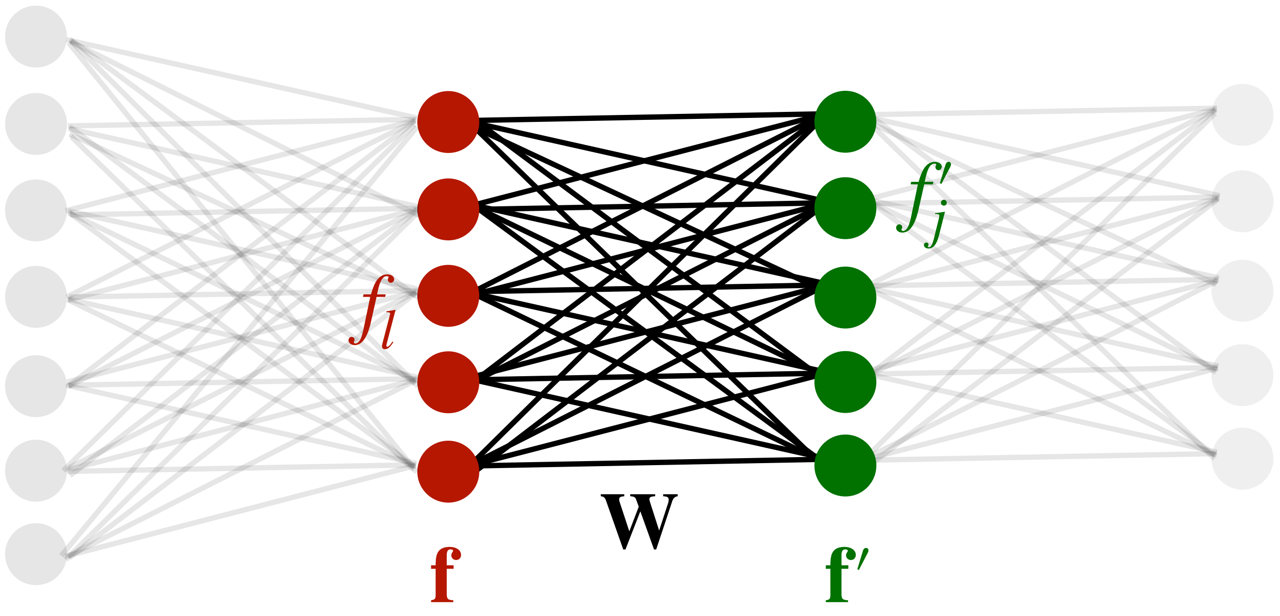
$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \qquad f'_j = \sum_l w^j_l(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

Convolutional message passing



$$\mathbf{m}_{ab} = \underbrace{\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)}_{\text{geometric attribute}}\,\mathbf{f}_b$$

Conditional linear layers



$\mathbf{f} \quad \mapsto \quad \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \quad \mapsto \quad \mathbf{f}'' = \sigma(\,\mathbf{f}'\,)$ (Repeat L times)

linear layer — activation

Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\ \mathbf{f} \qquad f'_j = \sum_l w_l^j f_l$$

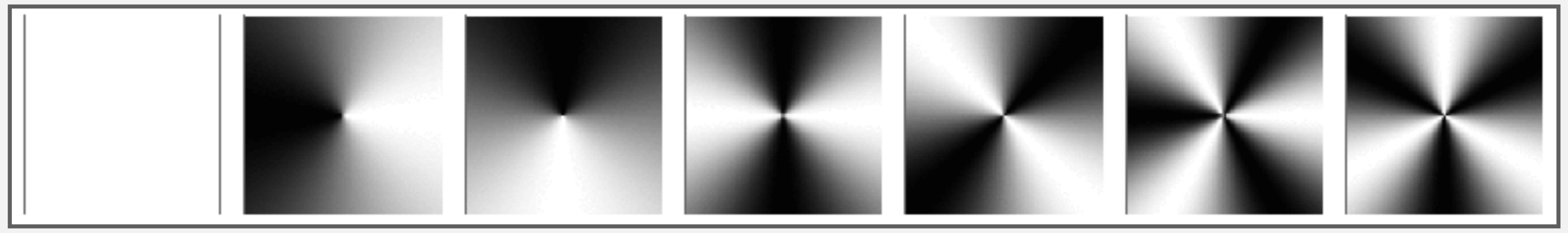
Conditional linear layer (weight matrix depends on $\mathbf{x}_b - \mathbf{x}_a$)

$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \qquad f'_j = \sum_l w_l^j(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

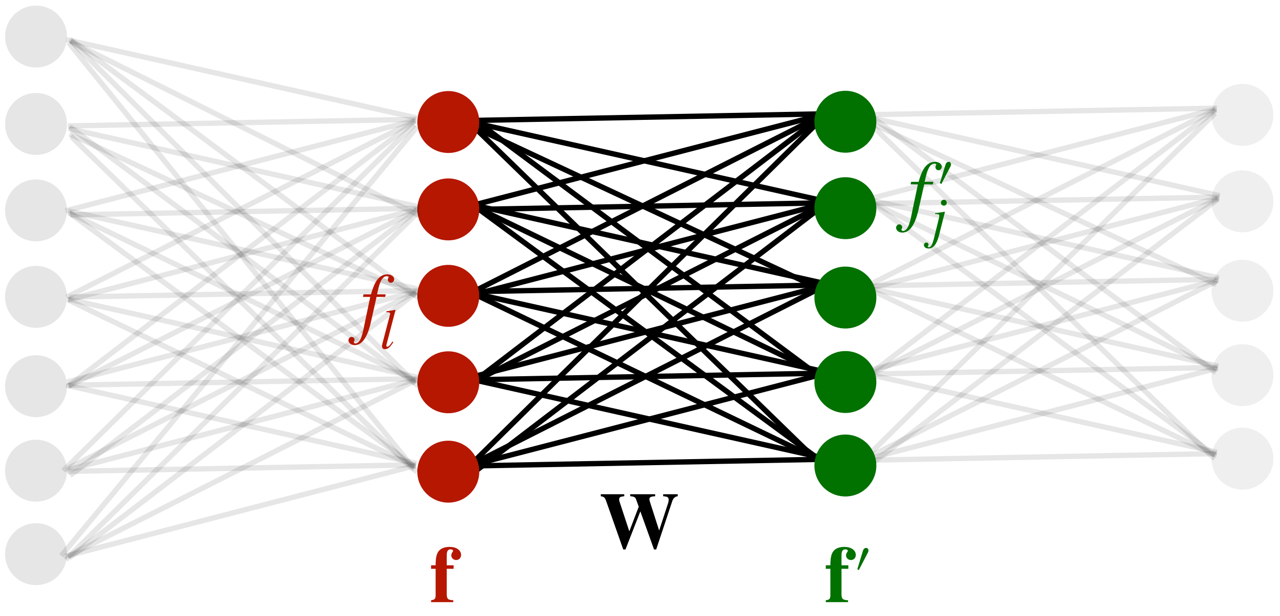
Let $\mathbf{W} : \mathbb{R}^3 \rightarrow \mathbb{R}^{C' \times C}$ a matrix valued function (conv-kernel)

- Expanded in a basis $Y(\mathbf{x}) = \begin{pmatrix} \vdots \\ Y_J(\mathbf{x}) \\ \vdots \end{pmatrix}$
- Basis (coordinate embedding) functions $Y_J : \mathbb{R}^3 \rightarrow \mathbb{R}$
- Matrix-valued weights $\mathbf{W}_J$ with elements w_{Jl}^j

$$\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a) = \sum_J \mathbf{W}_J\, Y_J(\mathbf{x}_b - \mathbf{x}_a)$$



Conditional linear layers



$\mathbf{f} \quad \mapsto \quad \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \quad \mapsto \quad \mathbf{f}'' = \sigma(\,\mathbf{f}'\,)$ (Repeat L times)

linear layer — activation

Linear layer (matrix-vector multiplication)

$$\mathbf{f}' = \mathbf{W}\;\mathbf{f} \qquad\qquad f'_j = \sum_l w^j_l\, f_l$$

Conditional linear layer (weight matrix depends on $\mathbf{x}_b - \mathbf{x}_a$)

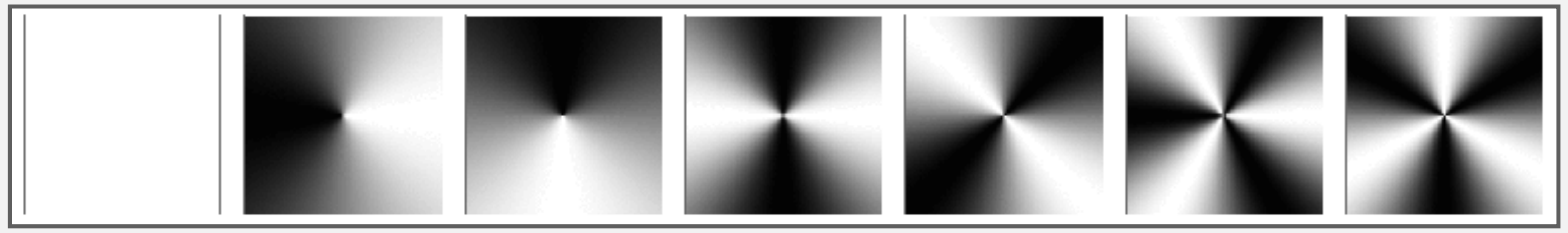
$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \qquad\qquad f'_j = \sum_l w^j_l(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

$$\mathbf{f}' = \mathbf{f} \overset{bilinear}{W} Y_J(\mathbf{x}_b - \mathbf{x}_a) \qquad\qquad f'_j = \sum_l \sum_J w^j_{Jl}\, Y_J(\mathbf{x}_b - \mathbf{x}_a)\, f_l$$

Let $\mathbf{W} : \mathbb{R}^3 \rightarrow \mathbb{R}^{C' \times C}$ a matrix valued function (conv-kernel)

- Expanded in a basis $Y(\mathbf{x}) = \begin{pmatrix} \vdots \\ Y_J(\mathbf{x}) \\ \vdots \end{pmatrix}$
- Basis (coordinate embedding) functions $Y_J : \mathbb{R}^3 \rightarrow \mathbb{R}$
- Matrix-valued weights $\mathbf{W}_J$ with elements w^j_{Jl}

$$\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a) = \sum_J \mathbf{W}_J\, Y_J(\mathbf{x}_b - \mathbf{x}_a)$$



Conditional linear layers

$\mathbf{f} \mapsto \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \mapsto \mathbf{f}'' = \sigma(\mathbf{f}')$

linear layer activation

Conditional linear layers are (partially evaluated) **tensor products!!!**

$$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \quad \Leftrightarrow \quad \mathbf{f}' = \mathbf{f} \otimes^{\mathbf{W}} Y_J(\mathbf{x}_b - \mathbf{x}_a)$$

$\mathbf{f}' = \mathbf{W}\,\mathbf{f}$

$f'_j = \sum_l w^j_l(\mathbf{x}_b - \mathbf{x}_a) f_l$

- Basis (coordinate embedding) functions $Y_J : \mathbb{R}^3 \to \mathbb{R}$
- Matrix-valued weights $\mathbf{W}_J$ with elements w^j_{Jl}

$$\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a) = \sum_J \mathbf{W}_J Y_J(\mathbf{x}_b - \mathbf{x}_a)$$

$$\mathbf{f}' = \mathbf{f} \overset{bilinear}{W} Y_J(\mathbf{x}_b - \mathbf{x}_a) \qquad f'_j = \sum_l \sum_J w^j_{Jl}\, Y_J(\mathbf{x}_b - \mathbf{x}_a) f_l$$

Conditional linear layers

$\mathbf{f} \mapsto \mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \mapsto \mathbf{f}'' = \sigma(\mathbf{f}')$ (Repeat layers)

linear layer activation

Linear layer (matrix-vector multiplication)

$\mathbf{f}' = \mathbf{W}\,\mathbf{f}$

Conditional linear layer

$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f}$ $f'_j = \sum_l w^j_l(\mathbf{x}_b - \mathbf{x}_a)\, f_l$

Matrix-valued function (conv-kernel)

Expressed in a basis $Y(\mathbf{x}) = \begin{pmatrix} \vdots \\ Y_J(\mathbf{x}) \\ \vdots \end{pmatrix}$

- Basis (coordinate embedding) functions $Y_J : \mathbb{R}^3 \to \mathbb{R}$
- Matrix-valued weights $\mathbf{W}_J$ with elements w^j_{Jl}

$\mathbf{W}(\mathbf{x}_b - \mathbf{x}_a) = \sum_J \mathbf{W}_J\, Y_J(\mathbf{x}_b - \mathbf{x}_a)$

$\mathbf{f}' = \mathbf{f} \overset{bilinear}{\mathbf{W}} Y_J(\mathbf{x}_b - \mathbf{x}_a)$ $f'_j = \sum_l \sum_J w^j_{Jl}\, Y_J(\mathbf{x}_b - \mathbf{x}_a)$

Conditional linear layers are (partially evaluated) **tensor products!!!**

$\mathbf{f}' = \mathbf{W}(\mathbf{x}_b - \mathbf{x}_a)\,\mathbf{f} \quad \Leftrightarrow \quad \mathbf{f}' = \mathbf{f} \otimes^{\mathbf{W}} Y_J(\mathbf{x}_b - \mathbf{x}_a)$

Next up Clebsch-Gordan tensor product:
A convenient TP for steerable vector spaces