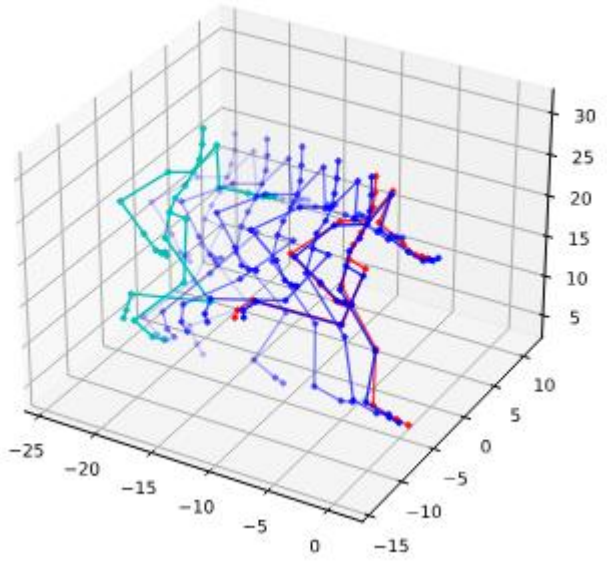
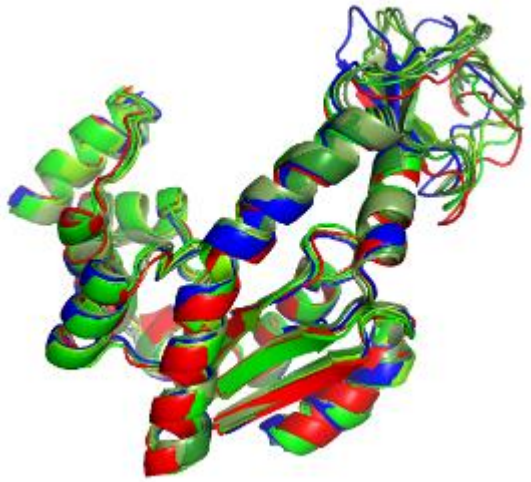




Equivariant Graph Neural Operator for Modeling 3D Dynamics

https://doi.org/10.48550/arXiv.2401.11037

Authors

Minkai Xu, Jiaqi Han, Aaron Lou, Jure Leskovec, Stefano Ermon
Stanford University

Jean Kossaifi, Kamyar Azizzadenesheli - *NVIDIA*

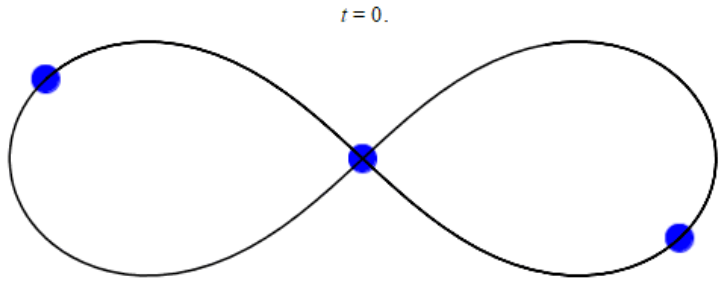
Arvind Ramanathan - *Argonne National Laboratory*

Anima Anandkumar - *California Institute of Technology*

Presentation

Eric Zander - *University of Oregon*

Three-Body Problem



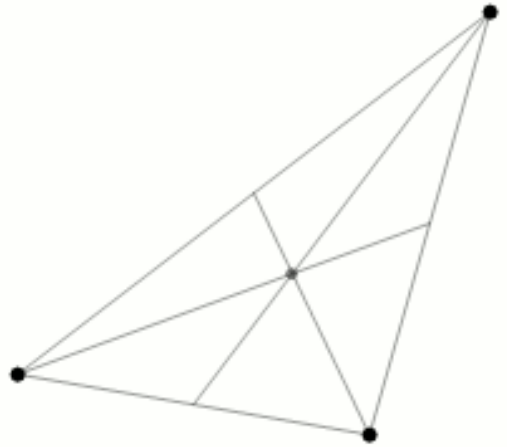
MaxwellMolecule, CC BY-SA 4.0 via Wikimedia Commons

Problem statement

Given three masses interacting through gravity, predict their motion over time

Conceptually simple, challenging in practice

- System is tightly coupled
- Sensitive to initial conditions
- Involves nonlinearity
- Numerical instability for close encounters
- No known analytical solution



Dnttllthmmnm, CC BY-SA 4.0 via Wikimedia Commons

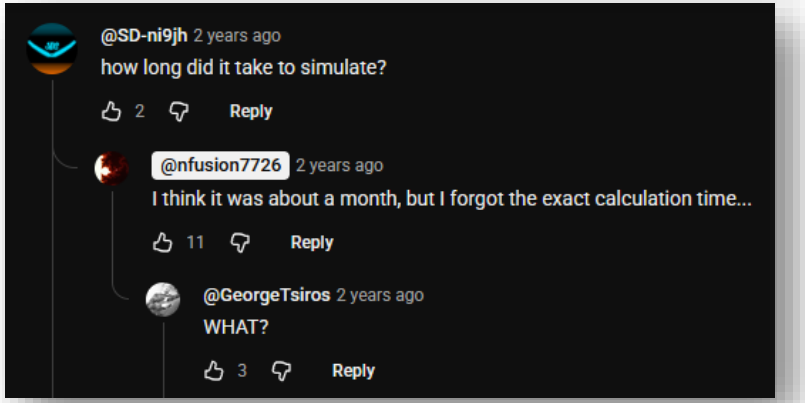
N-Body Problem

Simulation and challenges

- Direct summation: $O(N^2)$
- Barnes-Hut: O(N log N)
  - Group into clusters with trees
  - Approximate bodies as one mass
- Fast Multipole Method (FMM): ~O(N)
  - Groups via series expansions

Limitations

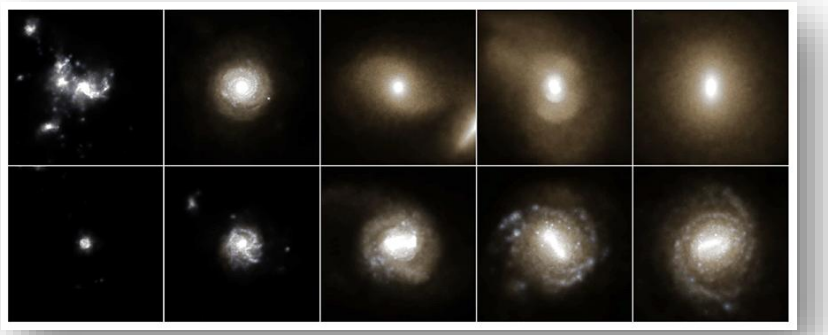
- Tradeoff between accuracy and speed
- Need small timesteps for close encounters



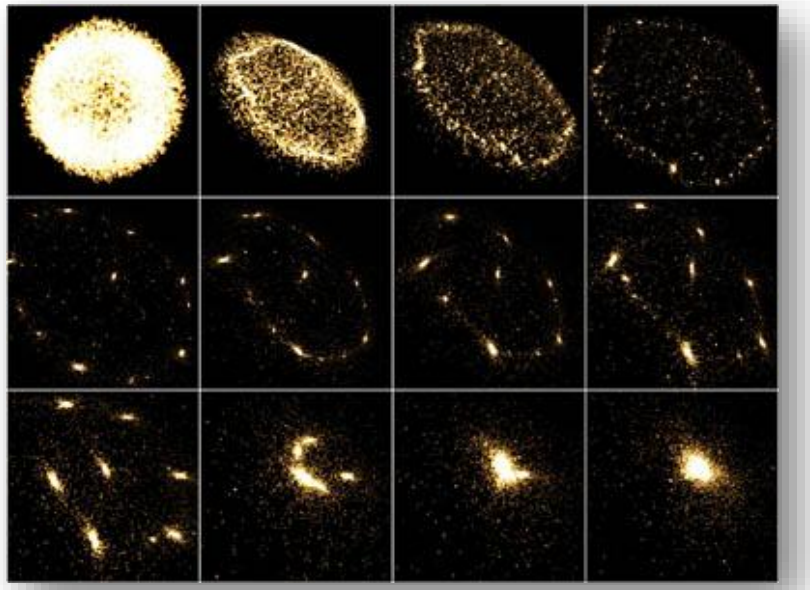
Particle-based Simulation Applications

Astrophysical systems

- N-body problem
- Galaxy formation
- Orbital mechanics

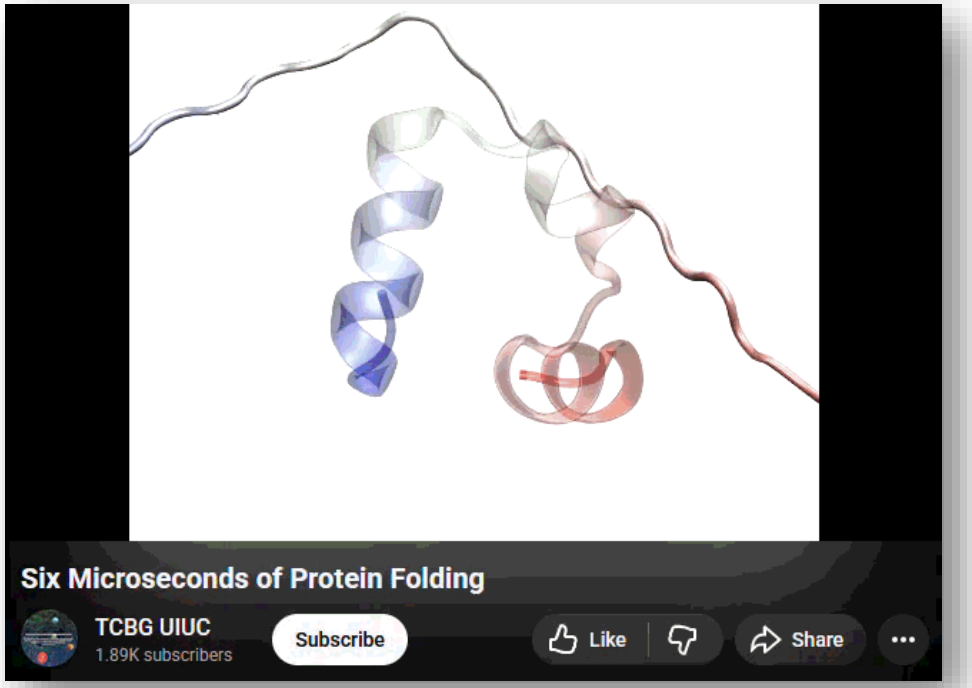


Simulated galaxy formation
Illustris Project



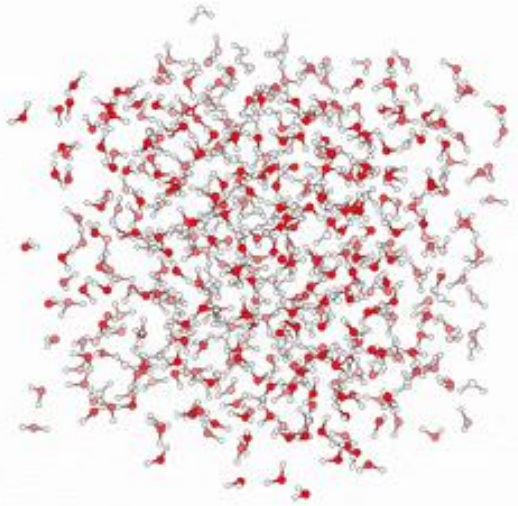
16,384-Body System,
Nyland et al., NVIDIA GPU Gems 3 Ch 31

Particle-based Simulation Applications



Molecular simulation

- Protein dynamics
- Drug discovery
- Material design



Some wiggling water
Kmckiern, CC BY-SA 4.0 via Wikimedia Commons

Particle-based Simulation Applications

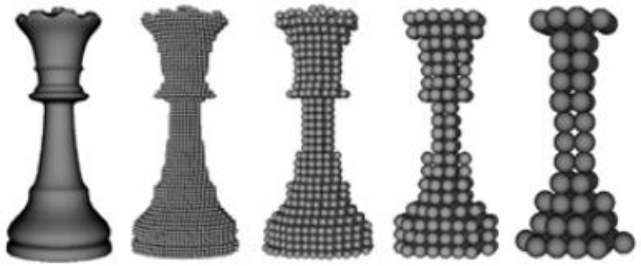


Figure 29-3 Particle Representations of a Chess Piece

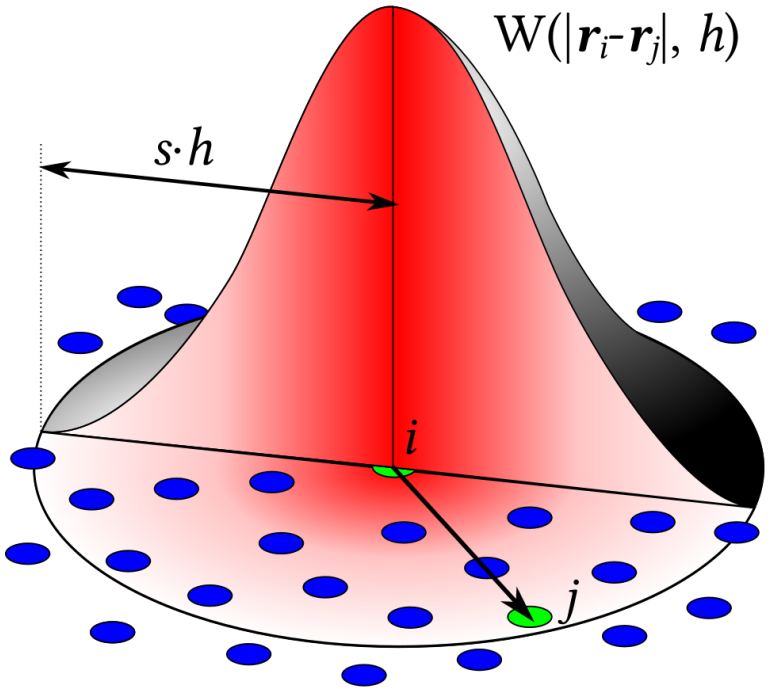
Harada, NVIDIA GPU Gems 3 Ch 29

"Teardown" by Tuxedo Labs

Rigid body mechanics

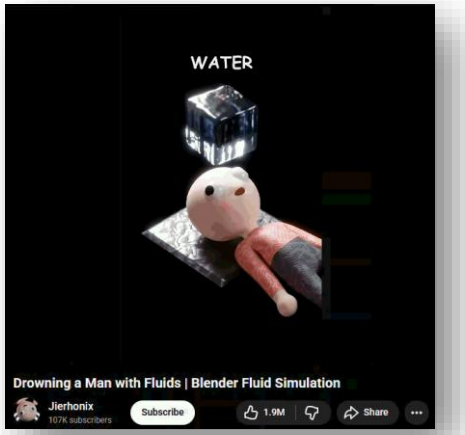
- Game engines
- Mechanical systems design
- Robotics simulation

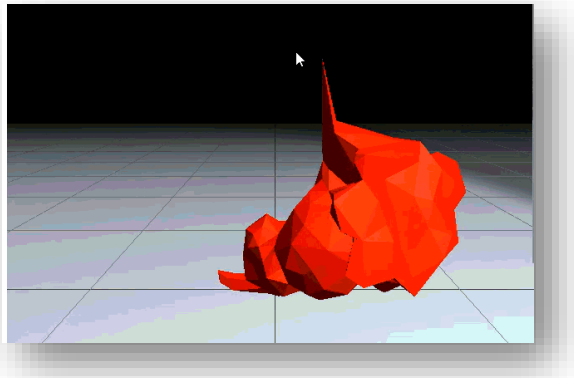
Particle-based Simulation Applications



Schematic view of a smoothed-particle hydrodynamics (SPH) convolution

Jlcercos, CC BY-SA 4.0 via Wikimedia Commons





Soft Body Simulation by Matthias Müller

Continuum mechanics

- Fluid/granular material sim
- Soft body simulation
- Engineering simulation

Particle-based Simulation Applications

Astrophysical systems

- N-body problem
- Galaxy formation
- Orbital mechanics

Molecular simulation

- Protein folding
- Drug discovery
- Material design

Rigid body mechanics

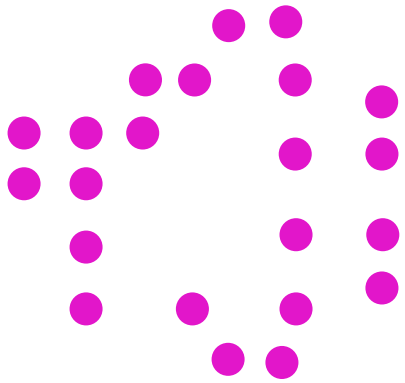
- Game engines
- Mechanical systems design
- Robotics simulation

Continuum mechanics

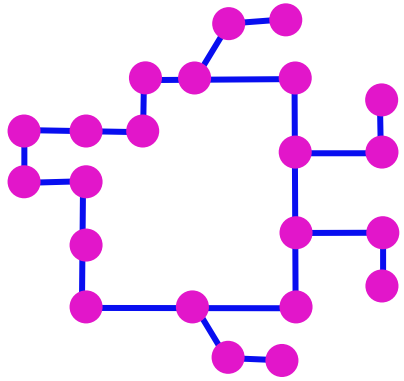
- Fluid/granular material sim
- Soft body simulation
- Engineering simulation

Graph-based Modeling

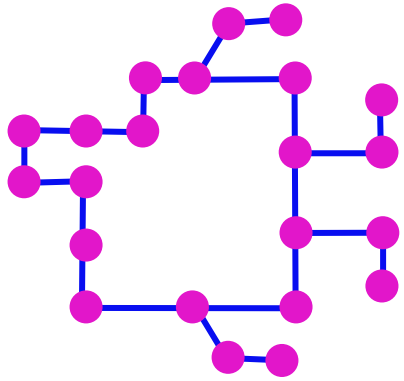
Graph-based Modeling



Graph-based Modeling

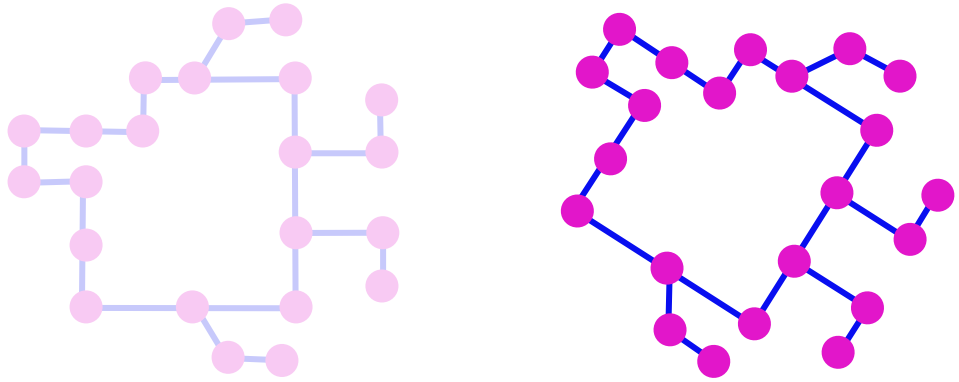


Graph-based Modeling



t = 0

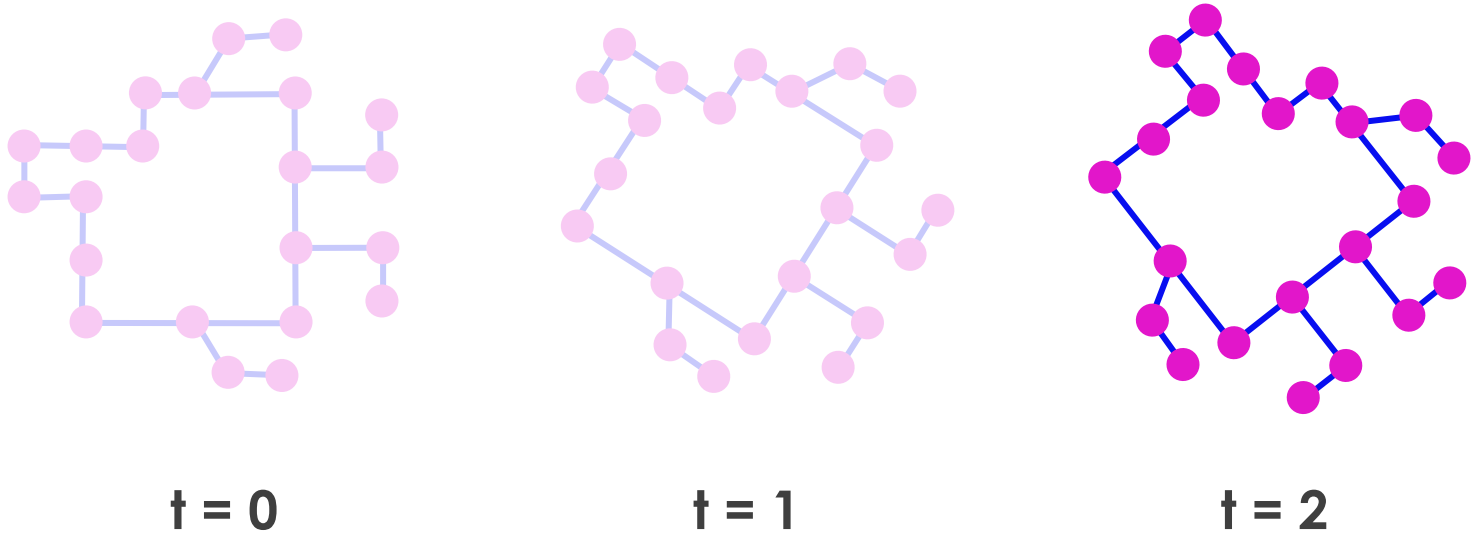
Graph-based Modeling



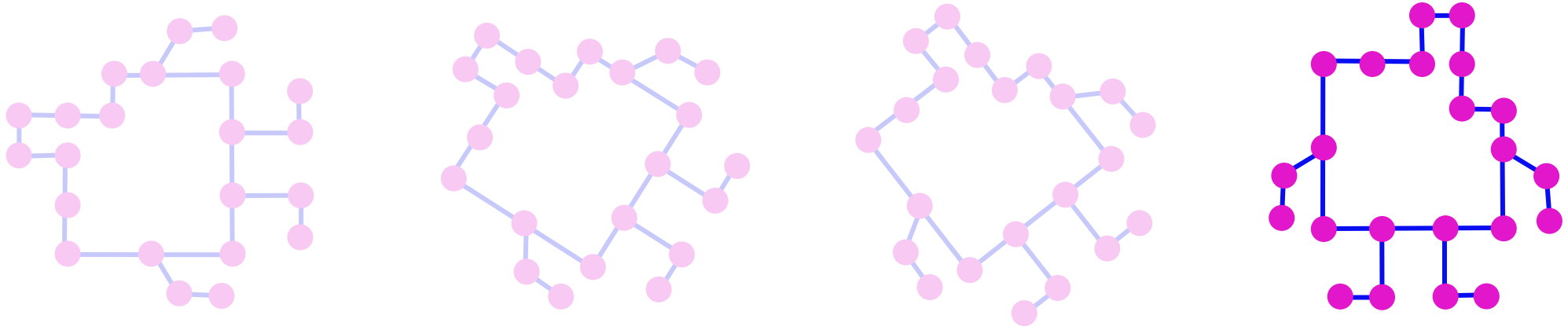
t = 0

t = 1

Graph-based Modeling

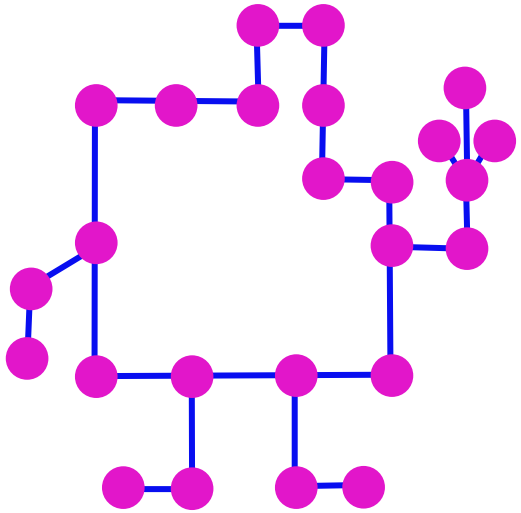


Graph-based Modeling



t = 0 **t = 1** **t = 2** **t = 3**

Graph-based Modeling



Geometric graph $G^{(t)}$

N particles as nodes

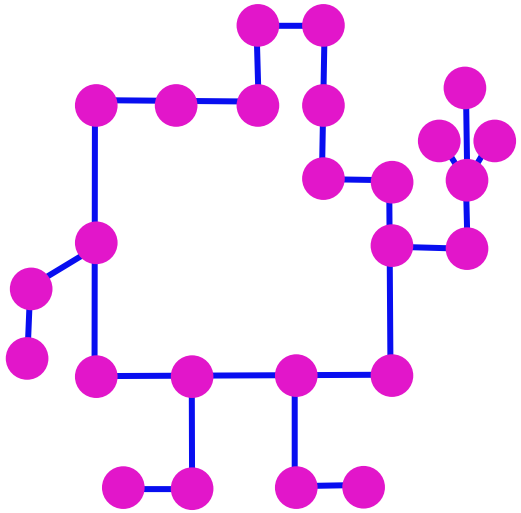
Node feature matrix $\boldsymbol{h} \in \mathbb{R}^{Nxk}$

Coordinate matrix $\boldsymbol{x} \in \mathbb{R}^{Nx3}$

Velocity matrix $\mathbf{v} \in \mathbb{R}^{Nx3}$

} Concatonated directional tensor $\mathbf{Z} \in \mathbb{R}^{Nx2x3}$

Graph-based Modeling



Geometric graph $G^{(t)}$

N particles as nodes

Node feature matrix $\boldsymbol{h} \in \mathbb{R}^{Nxk}$

Coordinate matrix $\boldsymbol{x} \in \mathbb{R}^{Nx3}$

Velocity matrix $\mathbf{v} \in \mathbb{R}^{Nx3}$

} Concatonated directional tensor $\mathbf{Z} \in \mathbb{R}^{Nx2x3}$

$\mathbf{Z}$ can be generalized to m dimensions if more than coordinates and velocity are needed. Xu et al. don't really discuss edge features or directed graphs.

Goal and Challenges

Goal

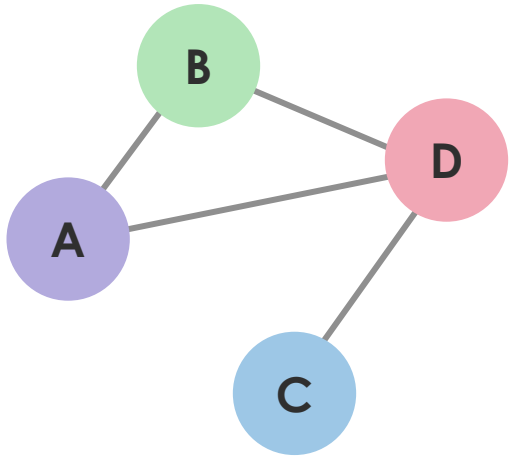
Given one or more geometric graphs $G^{(t)}$ in a sequence with node feature matrices $\boldsymbol{h}$ and m-directional tensors $\boldsymbol{Z}$, predict future states.

Challenges

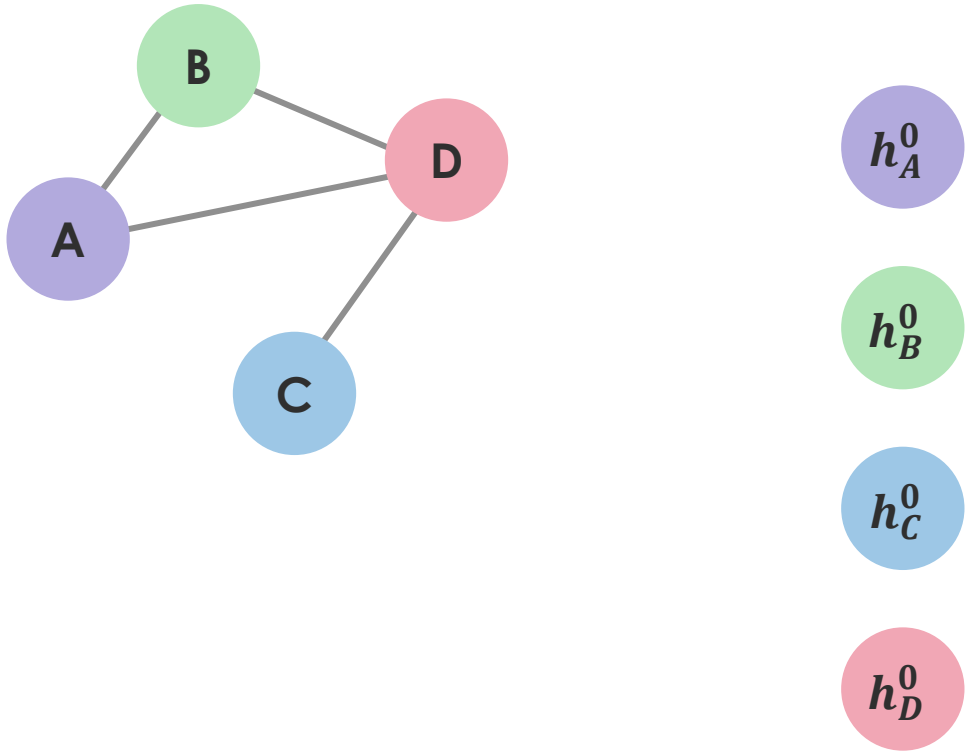
- How can we make use of graphs' structure?
- How can we respect temporal correlations?
- How can we support data efficiency?
- How can we support generalization?

Preliminaries & Related Work

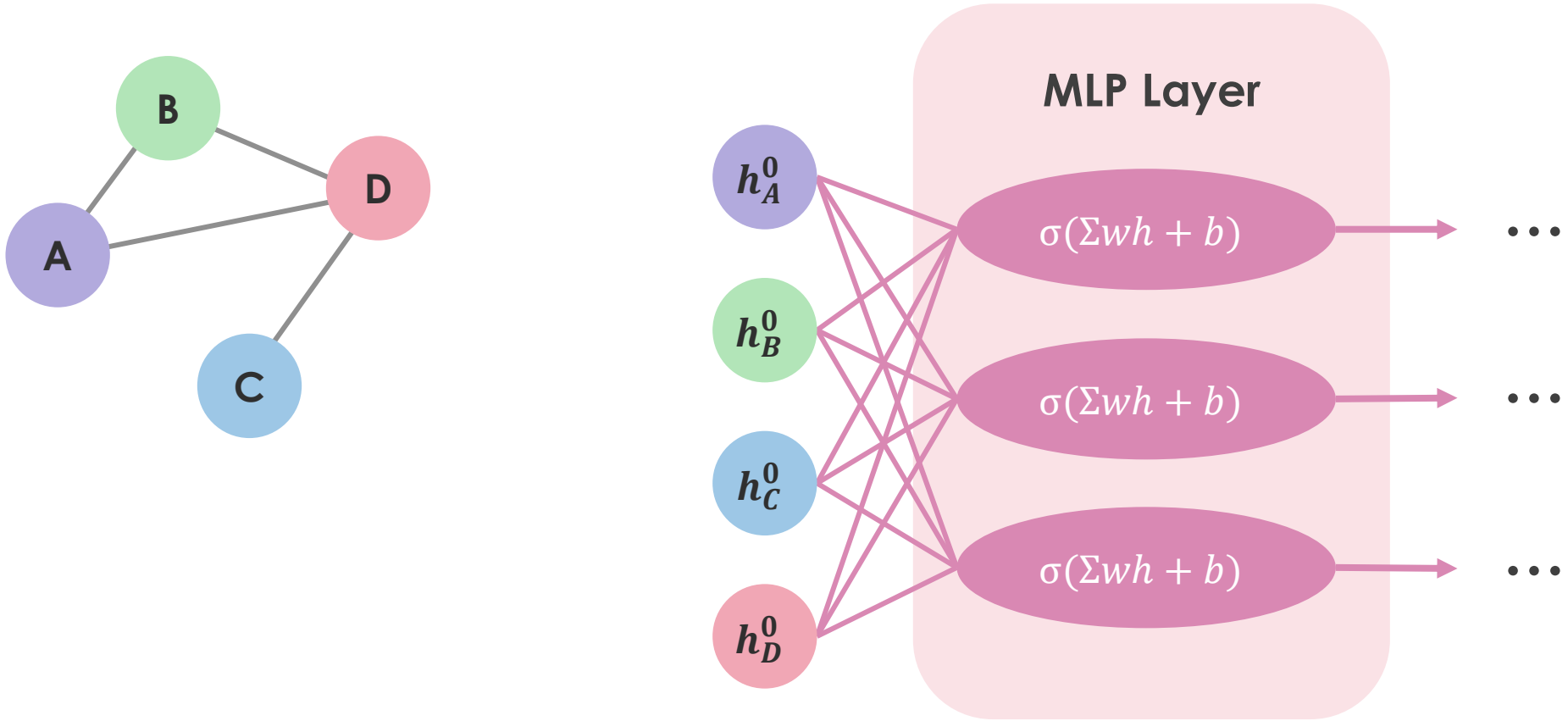
Graph Neural Networks



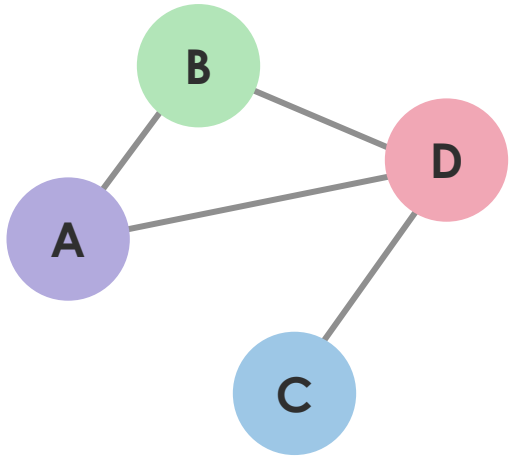
Graph Neural Networks



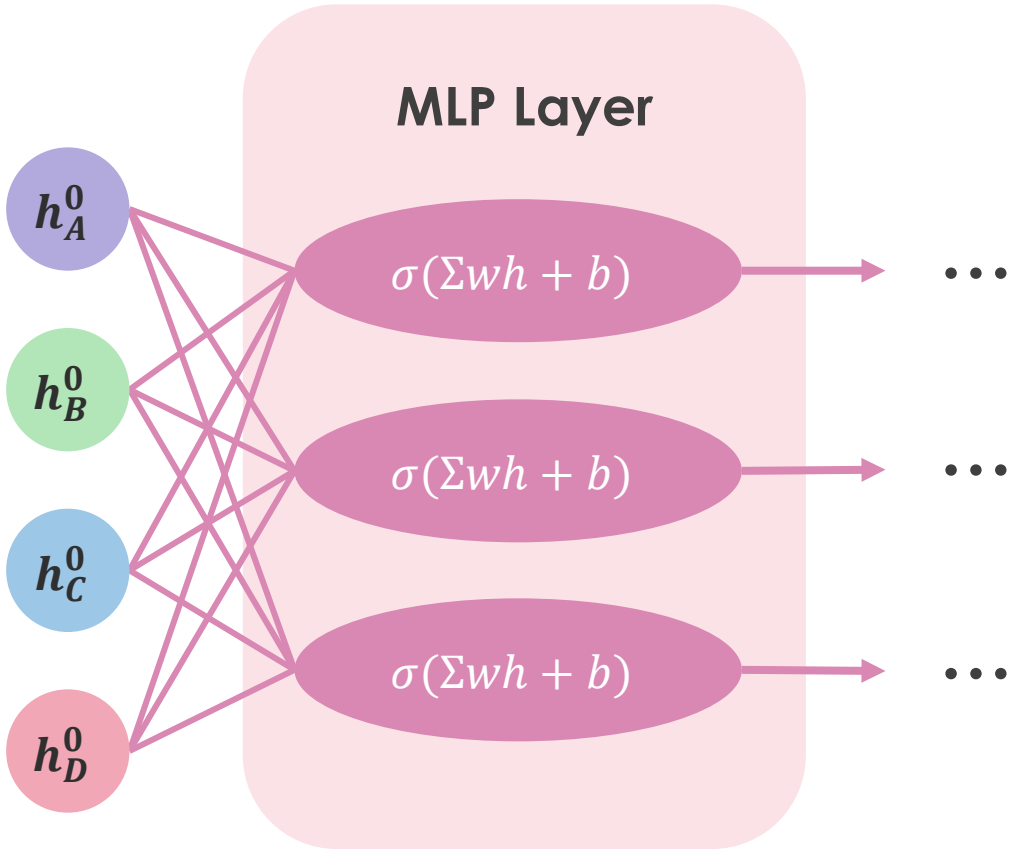
Graph Neural Networks



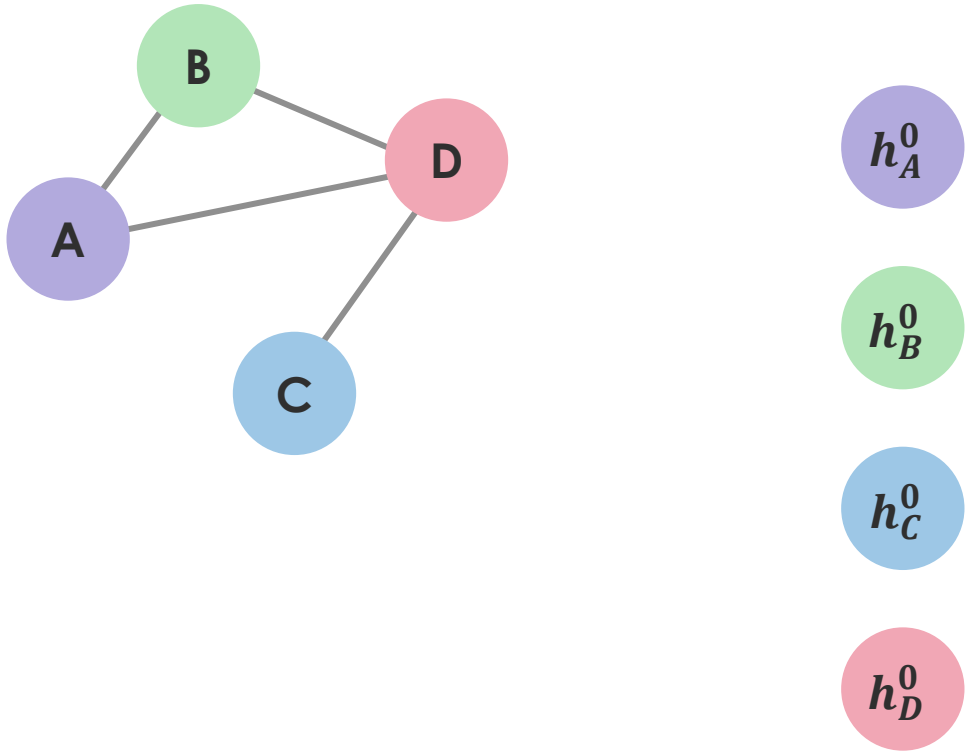
Graph Neural Networks



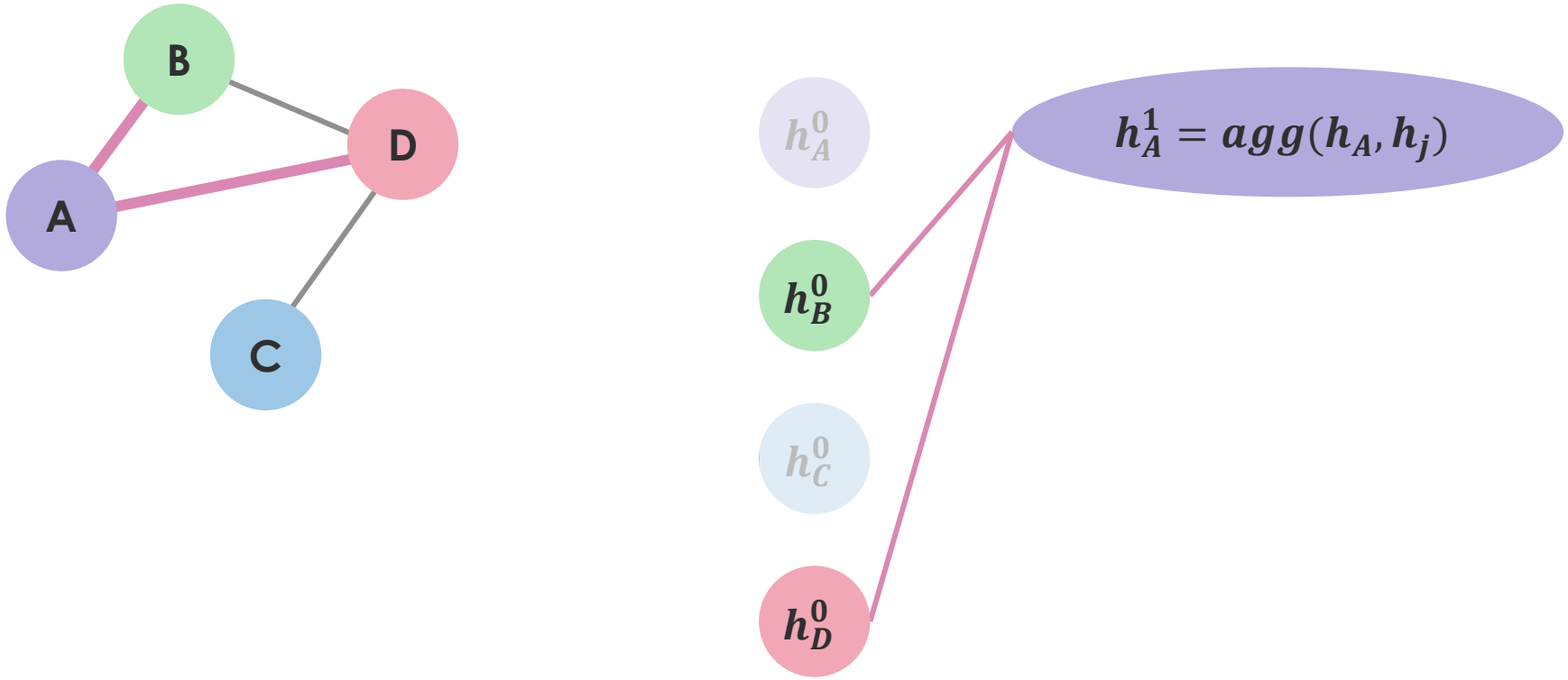
A traditional multi-layer perceptron doesn't (efficiently) incorporate structure!



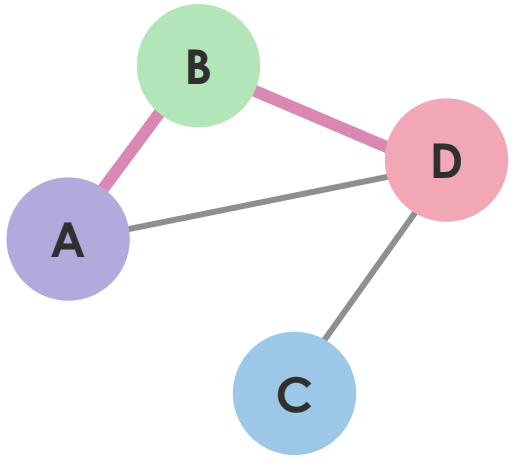
Graph Neural Networks

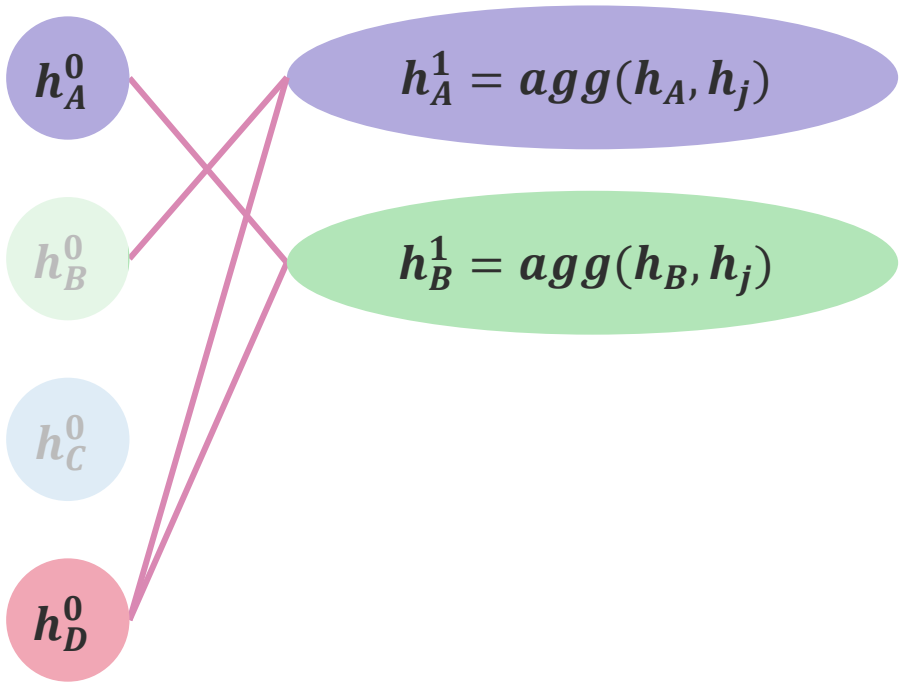


Graph Neural Networks

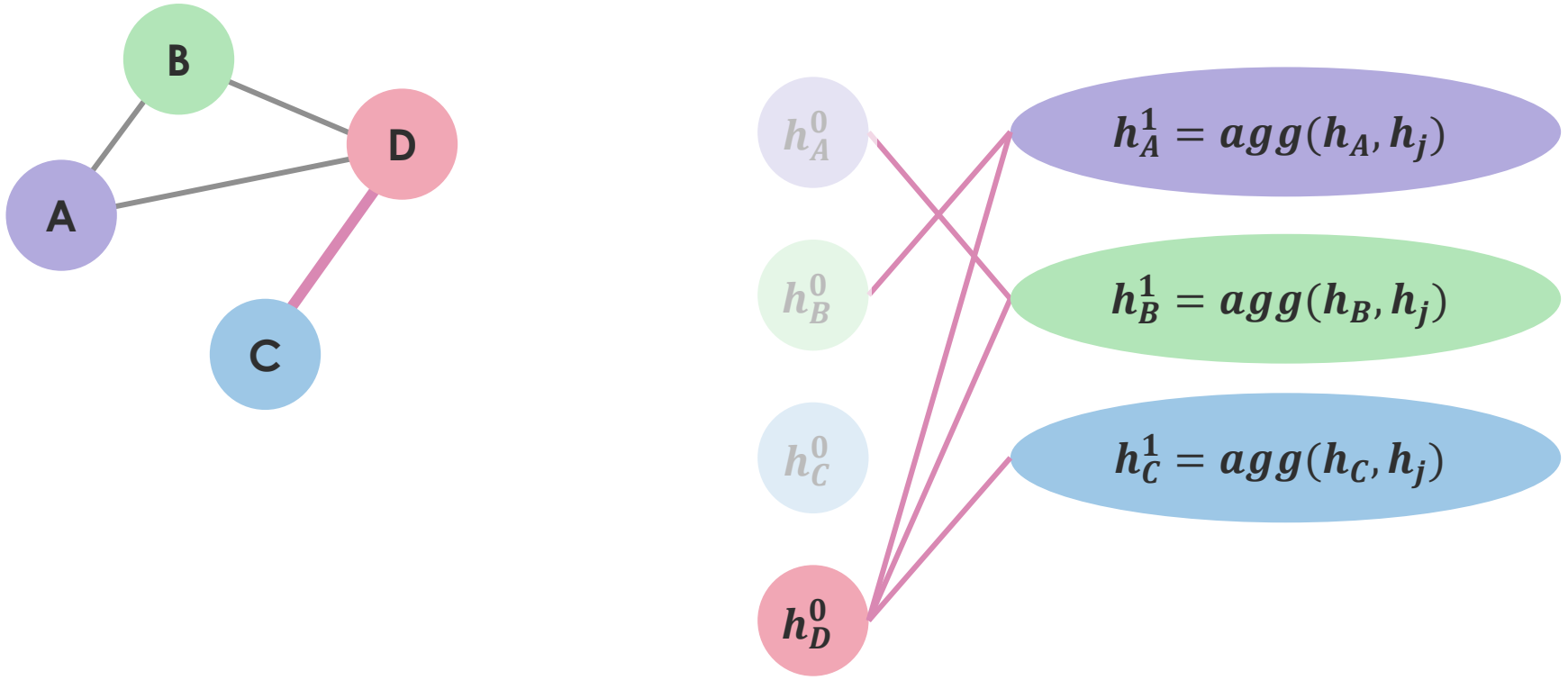


Graph Neural Networks

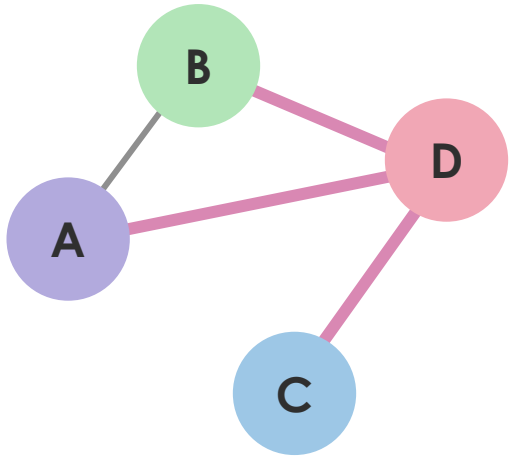


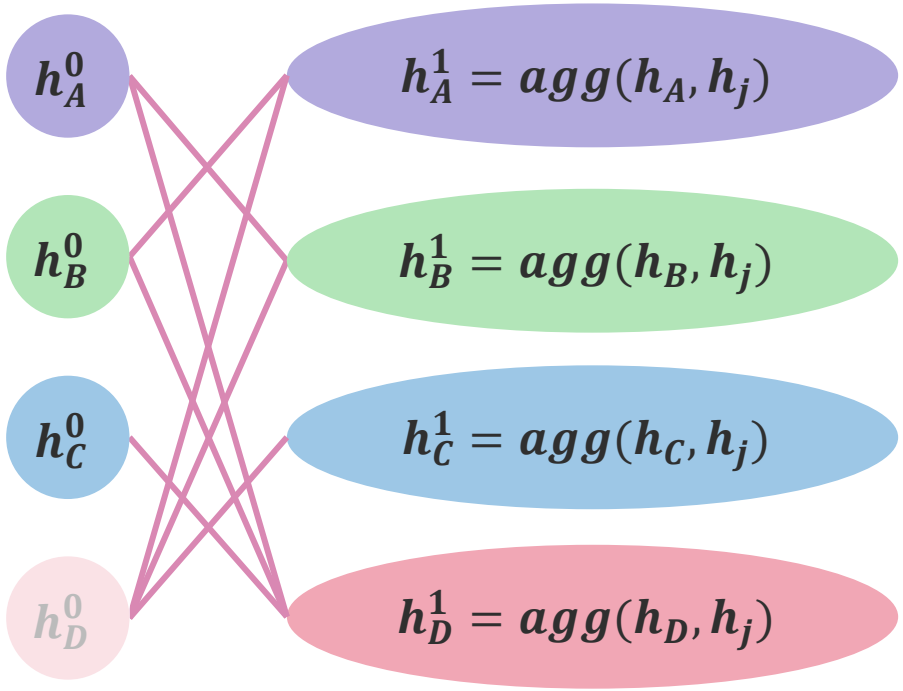


Graph Neural Networks



Graph Neural Networks



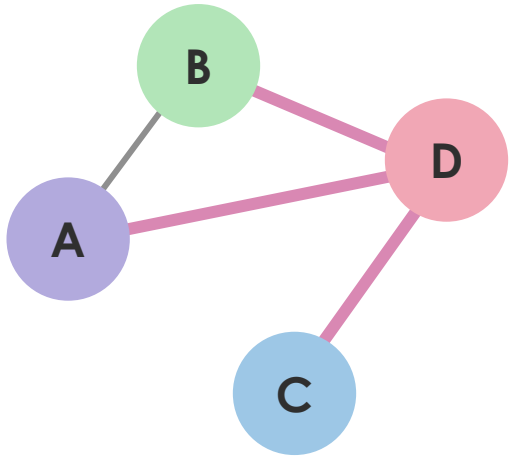


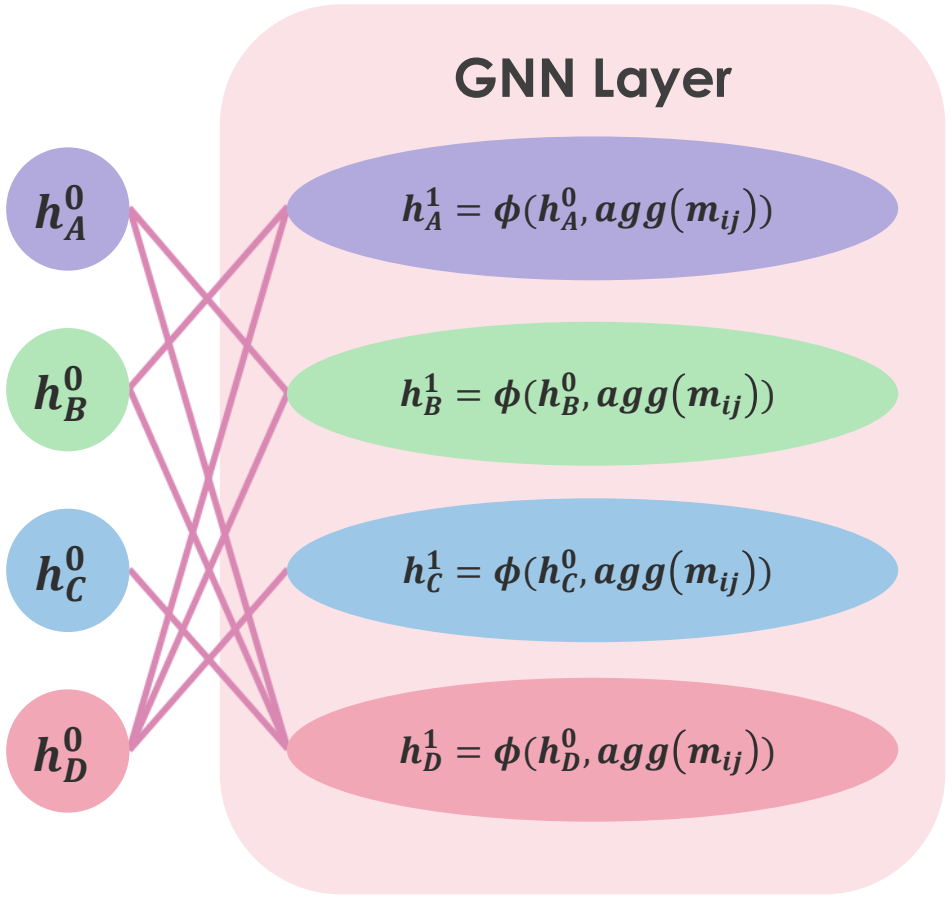
Graph Neural Networks

Message Passing!

$$m_{ij} = \phi_m(h_i, h_j)$$

$$h_i^{l+1} = \phi_h(h_i, agg(m_{ij}))$$

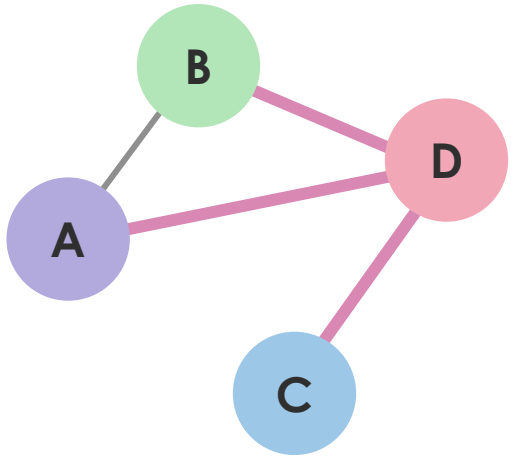




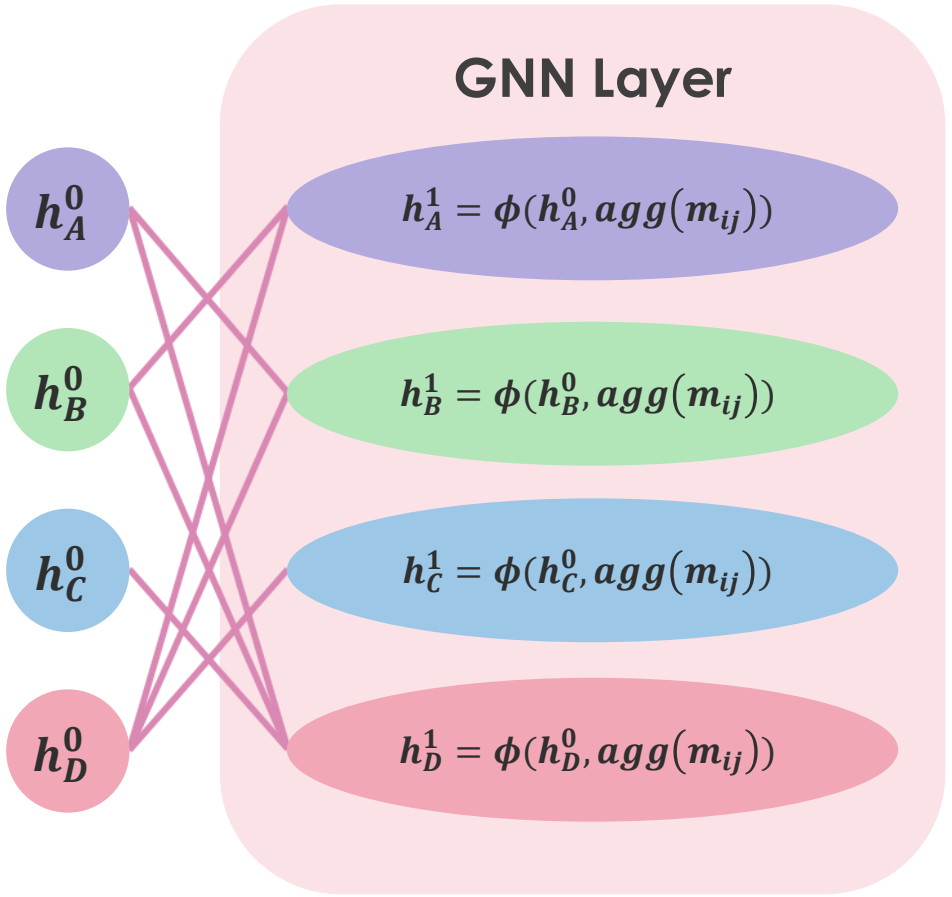
Graph Neural Networks

Message Passing!

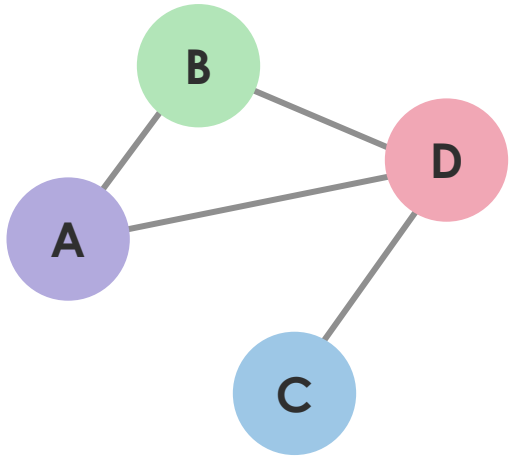
$$m_{ij} = \phi_m(h_i, h_j)$$
$$h_i^{l+1} = \phi_h(h_i, agg(m_{ij}))$$



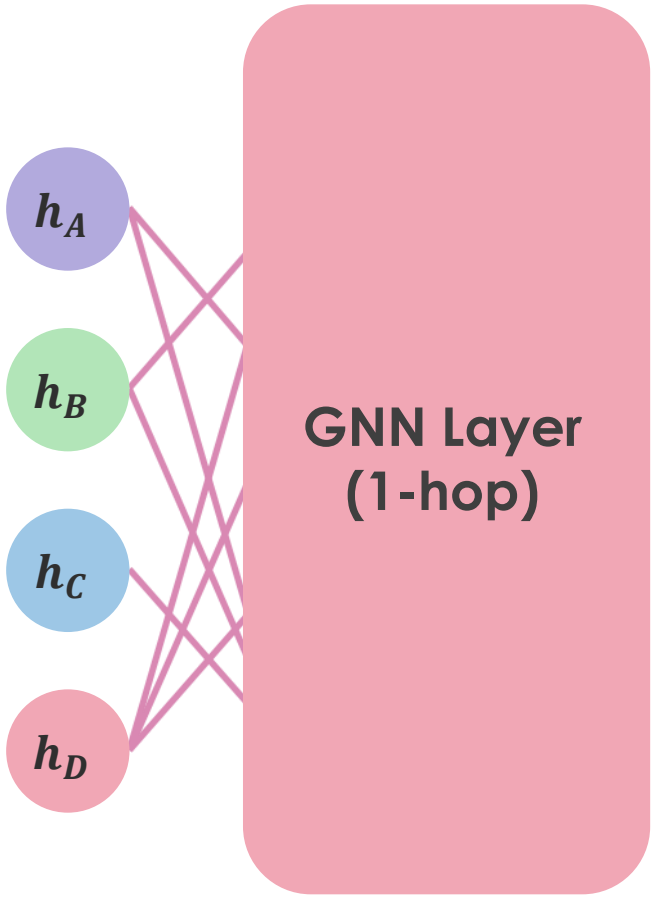
Different methods of aggregation are chosen for different techniques (e.g. GCNs and GATs)



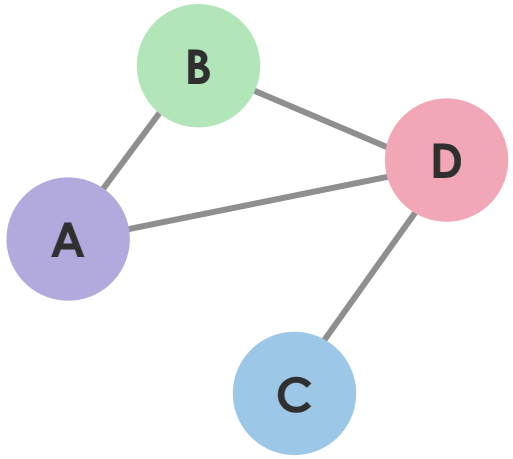
Graph Neural Networks



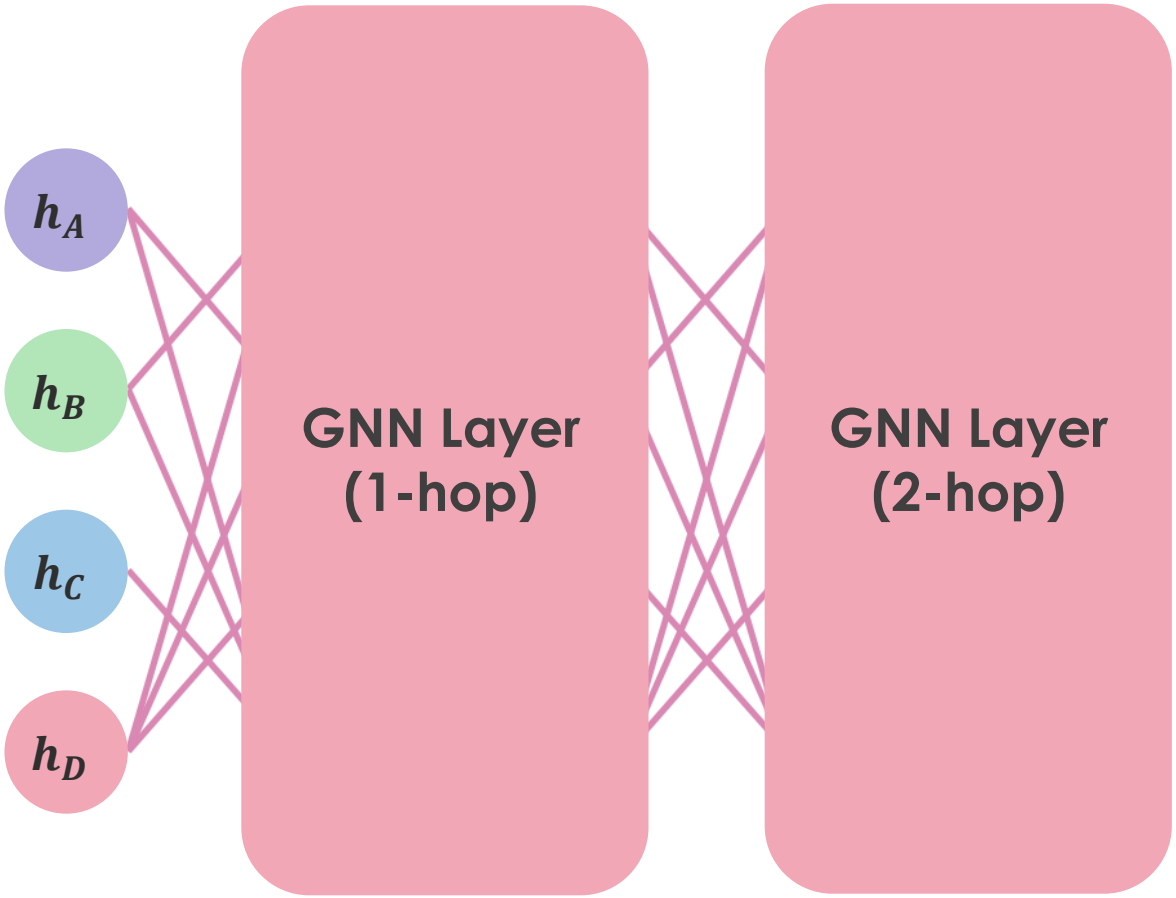
Different methods of aggregation are chosen for different techniques (e.g. GCNs and GATs)



Graph Neural Networks



Different methods of aggregation are chosen for different techniques (e.g. GCNs and GATs)



n GNN layers aggregate n-hop neighbors for each node

Addressing Two Concerns

- Equivariance
- Temporal Correlation

Equivariance

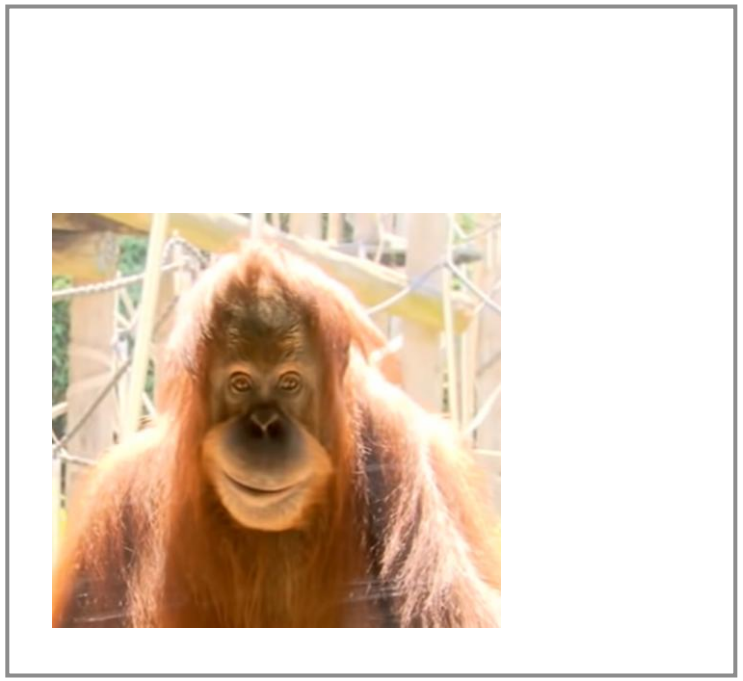


Image 1

Image 2

Equivariance

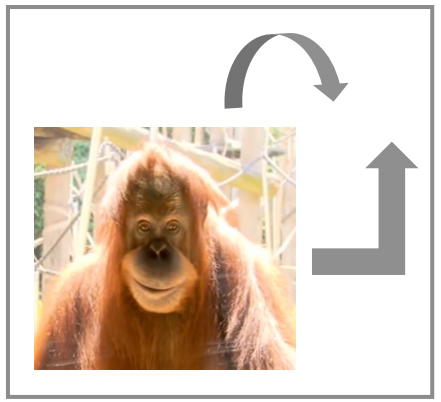


Image 1

Image 2

A model is **invariant** to transformations in the **Special Euclidean group SE(2)** if applying rotations or translations to the input does not change the output.

i.e. **transform input -> same output**

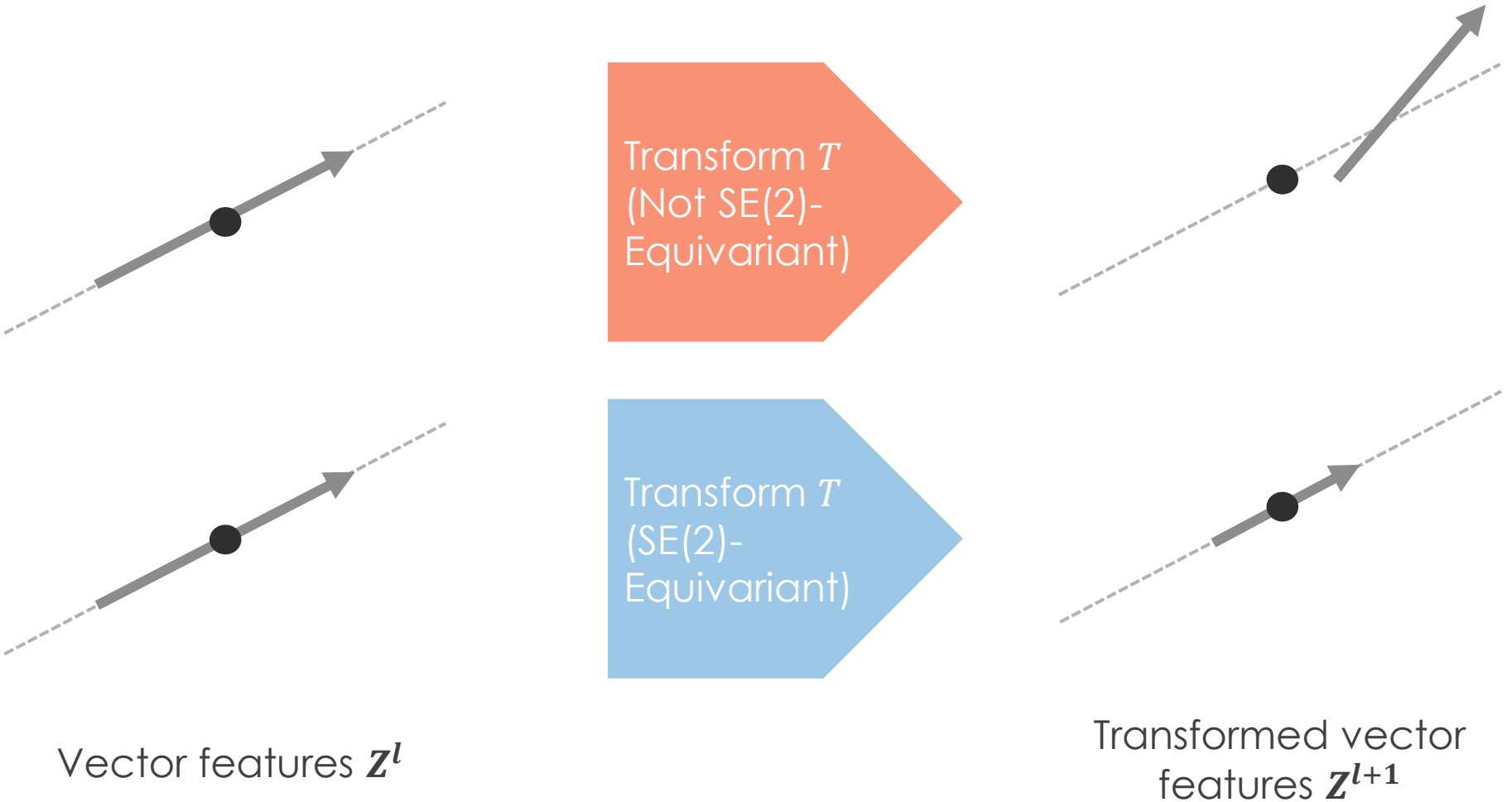
Equivariant models behave differently; rather than be agnostic to such transformations, they acknowledge and predictably model them.

i.e. **transform input -> transforms output the same way**

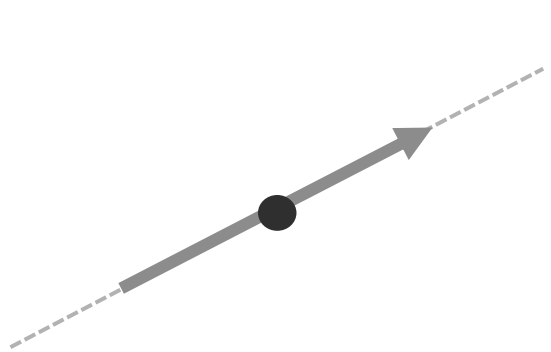
If f is the model and T is the transformation in the group SE(2), then the following describes equivariance:

$$f(T(x)) = T(f(x))$$

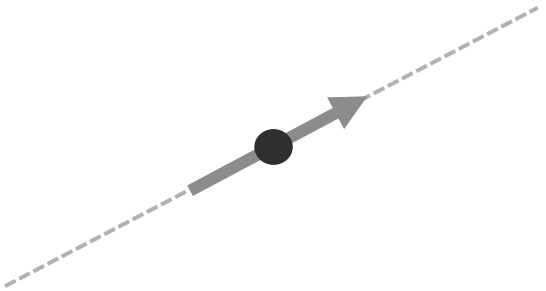
Equivariance



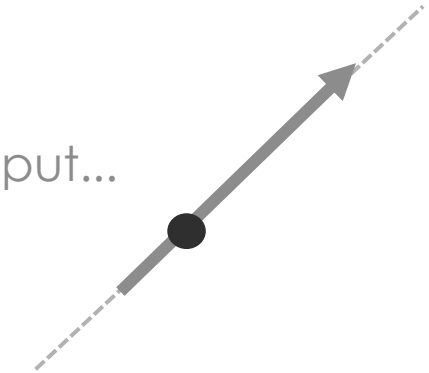
Equivariance



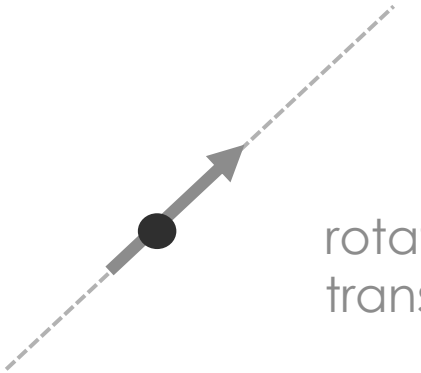
Transform T (SE(2)-Equivariant)



rotate and translate input...



Transform T (SE(2)-Equivariant)



rotate and translate output!

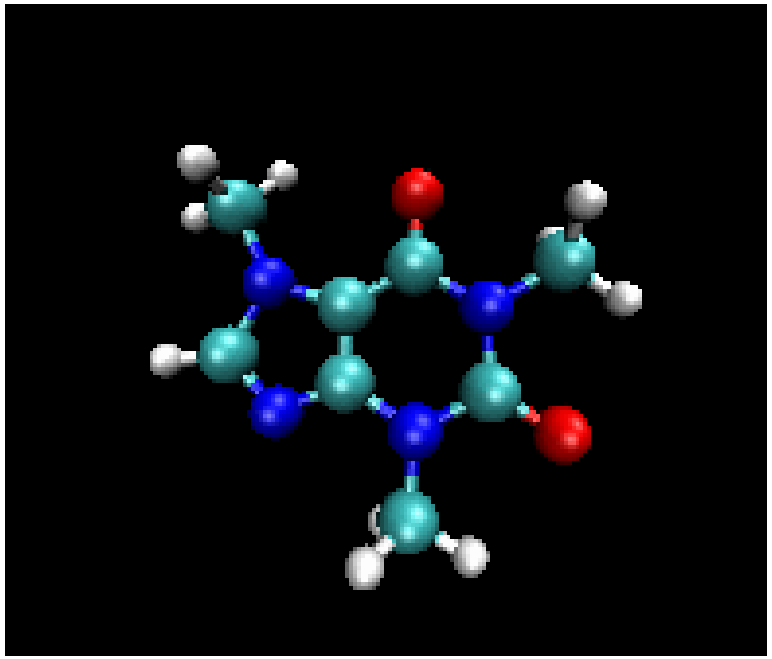
Vector features $\boldsymbol{Z^l}$

Transformed vector features $\boldsymbol{Z^{l+1}}$

Equivariance and SE(3)

Ideally, predicted positions $\boldsymbol{x}$ and velocities $\boldsymbol{v}$ for future times t can consistently build on rotations and translations made to prior states due to preserved directionality of vectors despite transformation.

Models that respect equivariance (i.e. are ***SE(3)-equivariant***) have shown improved data efficiency and generalization.



Bane of Michael

Related Work: Equivariance

GNS: Graph network-based simulators
TFN: Tensor field networks
EGNN: Equivariant GNN
SMP: Spherical message passing

- Translational equivariance
  - Clipping to translational radius (ex. GNS, Sanchez-Gonzalez et al., 2020)
  - Conv. filters from local point coordinates (ex. PointConv, Wu et al., 2020)
- Rotational equivariance
  - Spherical harmonics (ex. TFNs, Thomas et al., 2018)
  - Spherical harmonics + attention (ex. SE(3)-Transformer, Fuchs et al., 2020)
  - Equivariant message passing (ex. EGNNs, Satorras et al., 2023)
- Architectural enhancements
  - Local frame construction (ex. SMP, Liu et al., 2022)

Related Work: Spherical Harmonics

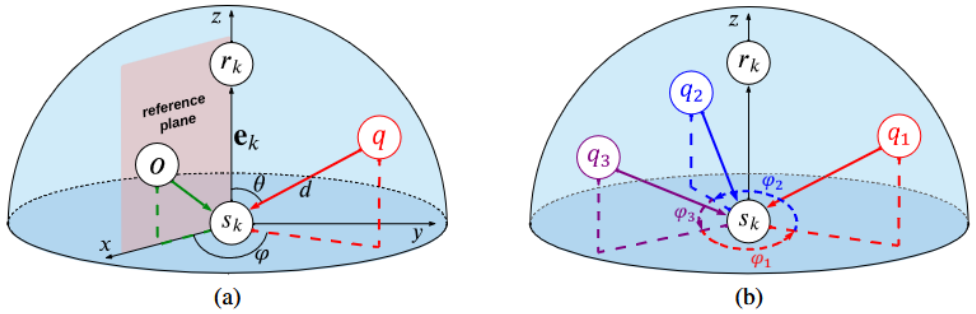
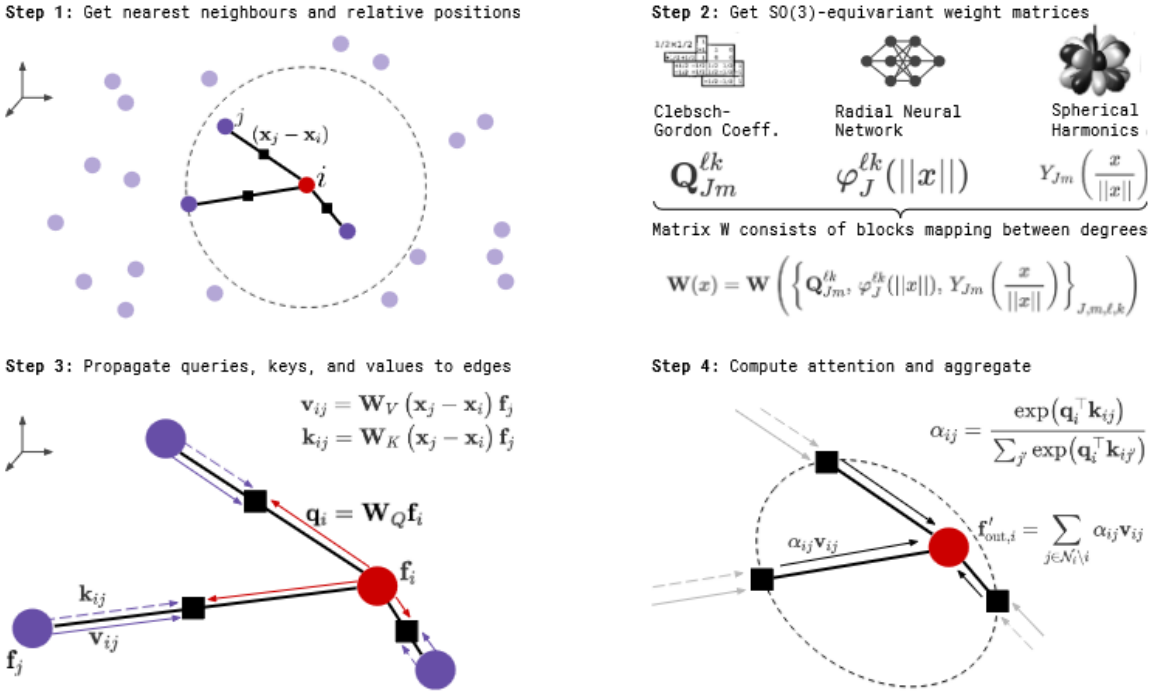


Diagram describing spherical message passing (SMP) wrt angles (a) and torsion (b) for molecular graphs.

Such representations encode directionalities and relative positions that transform predictably under rotation.

Liu et al., 2022



The SE(3)-Transformer describes nodes in terms of spherical harmonics (ie rotation-aware basis for functions on the sphere with angles θ & φ).

Rotating the input rotates the basis functions in predictable linear ways, allowing attention and feature updates to support equivariance by construction.

Fuchs et al., 2020

Related Work: EGNNs

m: *messages*
h: *node embedding*
x: *position coordinates*
a: *edge attributes (ex. positions x)*
l: *layer*
C: *local normalization factor*

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$$
$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$$

Graph convolutional layer (annotated)
Gilmer et al., 2017

Related Work: EGNNs

m: messages
h: node embedding
x: position coordinates
a: edge attributes (ex. positions x)
l: layer
C: local normalization factor

MLPs are not rotation-aware, and processing a directly (e.g. positions x) does not support rotational equivariance

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$$
$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$$

Graph convolutional layer (annotated)
Gilmer et al., 2017

Related Work: EGNNs

m: messages
h: node embedding
x: position coordinates
a: edge attributes (ex. positions x)
l: layer
C: local normalization factor

MLPs are not rotation-aware, and processing a directly (e.g. positions x) does not support rotational equivariance

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$$
$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$$

Graph convolutional layer (annotated)
Gilmer et al., 2017

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$
$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)$$
$$\mathbf{m}_i = \sum_{j \neq i} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$

Equivariant graph convolutional layer (annotated)
Satorras et al., 2021

Related Work: EGNNs

m: messages
h: node embedding
x: position coordinates
a: edge attributes (ex. positions x)
l: layer
C: local normalization factor

MLPs are not rotation-aware, and processing a directly (e.g. positions x) does not support rotational equivariance

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$$
$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$$

Graph convolutional layer (annotated)
Gilmer et al., 2017

Distances are rotation-invariant: fair game for use as MLP input

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$
$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)$$
$$\mathbf{m}_i = \sum_{j \neq i} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$

Equivariant graph convolutional layer (annotated)
Satorras et al., 2021

Related Work: EGNNs

m: messages
h: node embedding
x: position coordinates
a: edge attributes (ex. positions x)
l: layer
C: local normalization factor

MLPs are not rotation-aware, and processing a directly (e.g. positions x) does not support rotational equivariance

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$$
$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$$

Graph convolutional layer (annotated)
Gilmer et al., 2017

Distances are rotation-invariant: fair game for use as MLP input

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$
$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C \sum_{j \neq i} \left(\mathbf{x}_i^l - \mathbf{x}_j^l\right) \phi_x\left(\mathbf{m}_{ij}\right)$$
$$\mathbf{m}_i = \sum_{j \neq i} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$

Equivariant structure update. Rotates position consistently based on relative vectors $x_i - x_j$, then stretches by learned scalars $\phi(\mathbf{m})$

Equivariant graph convolutional layer (annotated)
Satorras et al., 2021

Related Work: EGNNs

m: messages
h: node embedding
x: position coordinates
a: edge attributes (ex. positions x)
l: layer
C: local normalization factor

MLPs are not rotation-aware, and processing a directly (e.g. positions x) does not support rotational equivariance

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$$
$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$$

Graph convolutional layer (annotated)
Gilmer et al., 2017

Distances are rotation-invariant: fair game for use as MLP input

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$
$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)$$
$$\mathbf{m}_i = \sum_{j \neq i} \mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$

Equivariant structure update. Rotates position consistently based on relative vectors $x_i - x_j$, then stretches by learned scalars $\phi(\mathbf{m})$

Equivariant graph convolutional layer (annotated)
Satorras et al., 2021

E. Sometimes invariant features are all you need. 😑

Temporal Correlation

Xu et al. argue that all these prior approaches help with spatial interactions, but fail to ideally model temporal correlations.

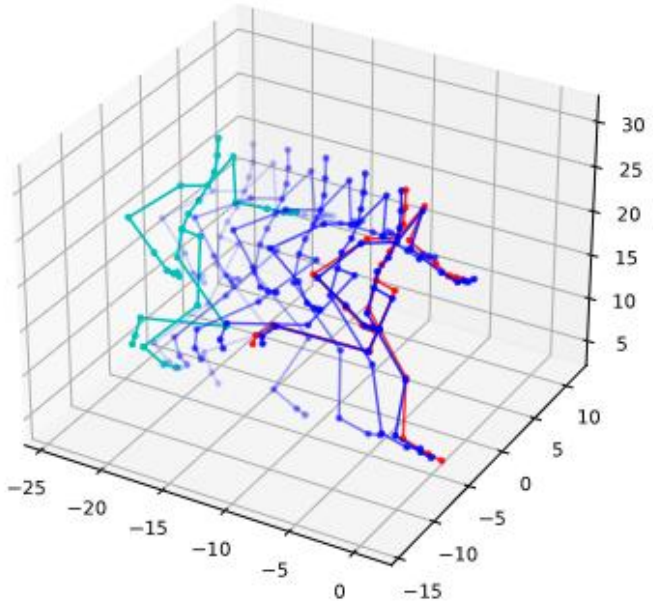
Temporal Correlation

Ideally, model learning respects that…

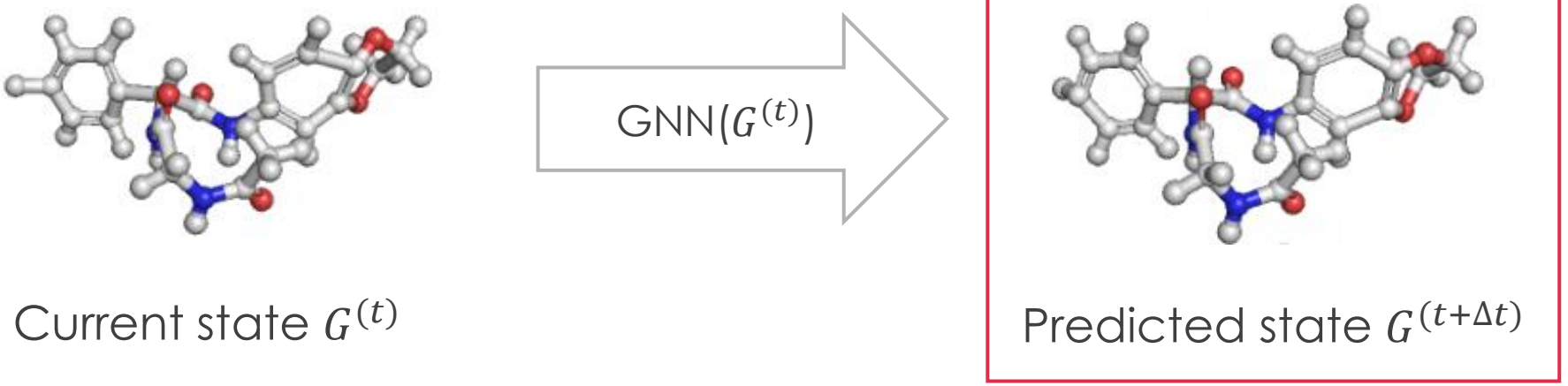
- Velocities persist
- Systems exhibit inertia
- Forces accumulate

But next frame prediction…

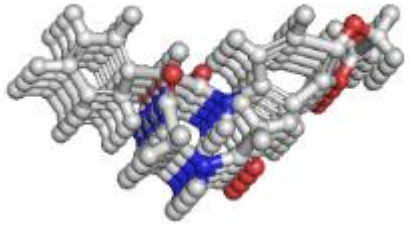
- Drifts due to accumulated error
- Learns interpolation, not dynamics
- Requires temporal discretization



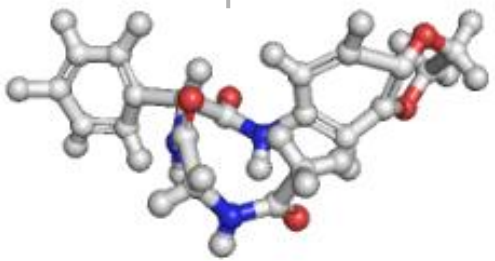
Neural Operators



Neural Operators



Input trajectory function is trivially the current state repeated for different values of Δt: $\{G^{(t)}\}$

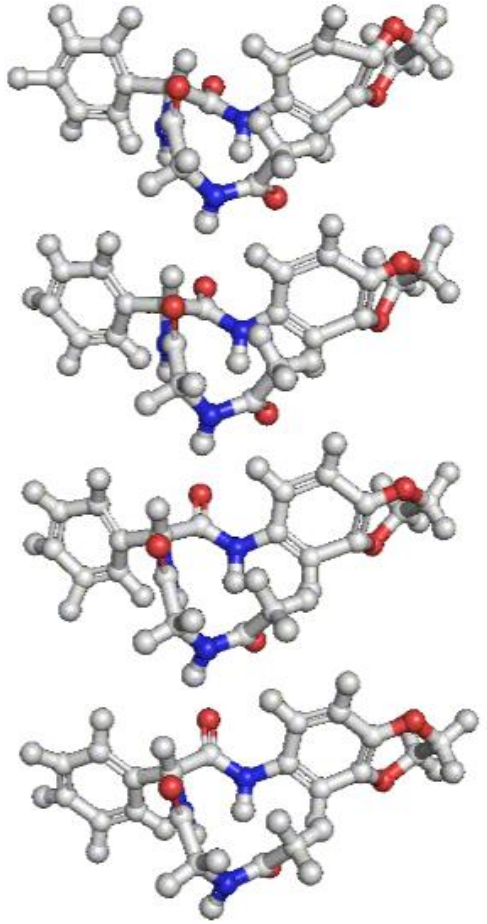


Current state: $G^{(t)}$

NO($G^{(t)}$)

Time embeddings: $t + \Delta t$

Function $f_G(t)$



Predicted trajectory function: $\{G^{(t+\Delta t)}\}$

Neural Operators

Unlike networks that map between finite Euclidean spaces or sets, neural operators approximate mappings between infinite-dimensional function spaces (and Banach spaces in particular)

- Neural operators more naturally support problems emphasizing function spaces
  - e.g. Differential equations
- They support discretization-invariance
  - e.g. Train on one discretization such as 32 points and apply on another such as 128 points. Not perfectly though, just structurally possible!

Banach spaces: Normed, complete vector spaces

i.e. Function spaces where distances are defined and convergence plays nice for optimization problems

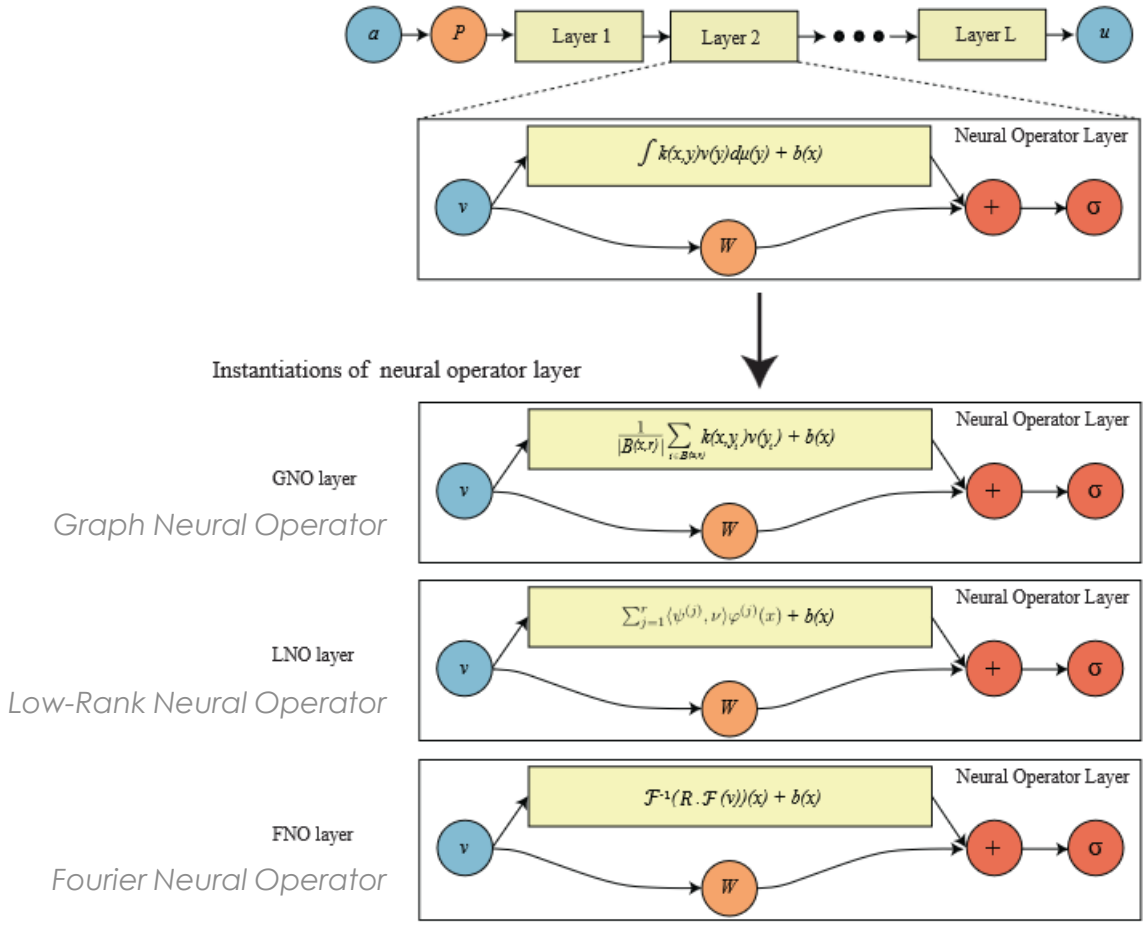


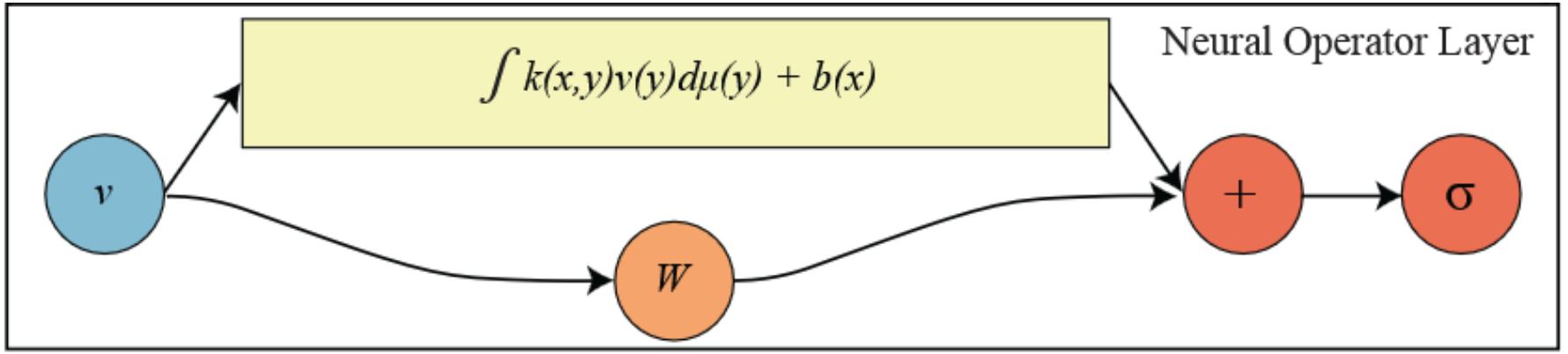
Figure 2: Neural operator architecture schematic
The input function a is passed to a pointwise lifting operator P that is followed by T layers of integral operators and pointwise non-linearity operations σ. In the end, the pointwise projection operator Q outputs the function u. Three instantiation of neural operator layers, GNO, LNO, and FNO are provided.

Kovachki et al., 2021

Neural Operators

The idealized operation is a weighted (k) aggregation of all points (y) for a point (x) with a local bias (b). Like a continuous, non-local generalization of a convolution.

Input function $v(y)$

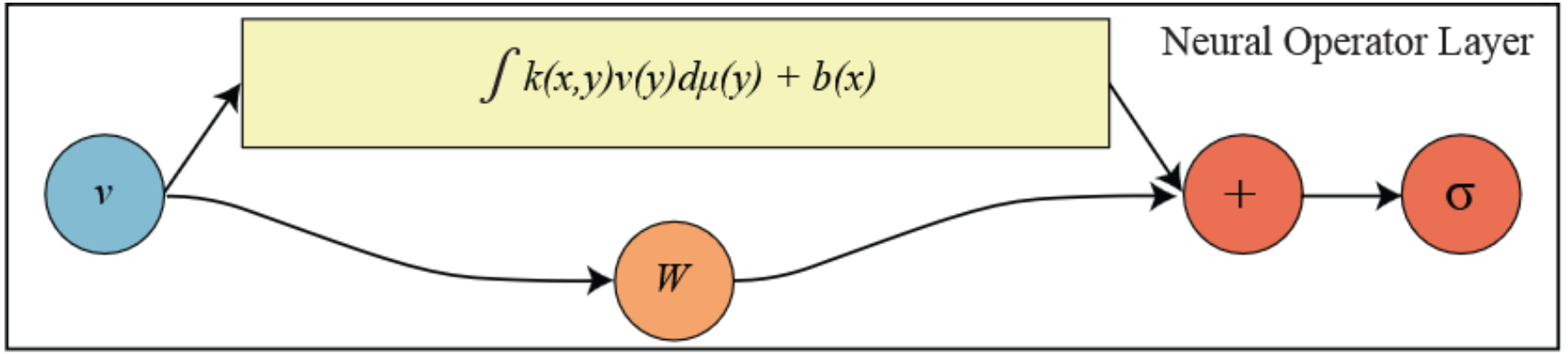


Neural Operators

The idealized operation is a weighted (k) aggregation of all points (y) for a point (x) with a local bias (b). Like a continuous, non-local generalization of a convolution.

However, in practice this is too damn expensive ($O(N^2)$ when discretized) and will explode your prized Dell Inspiron 4100

Input function $v(y)$



Frequency Domain

To practically achieve similar behavior, we can use an FFT F, retain only valuable low frequency modes I for efficiency, perform a convolution by multiplying in the frequency domain, and then use an inverse FFT F^{-1}.

This convolution is spatially non-local and works due to the principles described by the convolution theorem.

Convolution Theorem

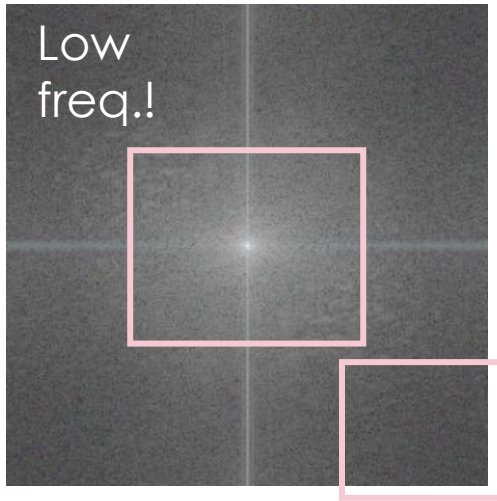
$$f(x,y) * h(x,y) \quad \Leftrightarrow \quad F(u,v)H(u,v)$$

Convolution in the spatial domain is equivalent to multiplication in the frequency domain

Spatial domain

$\mathcal{F}^{-1}$

$\mathcal{F}$

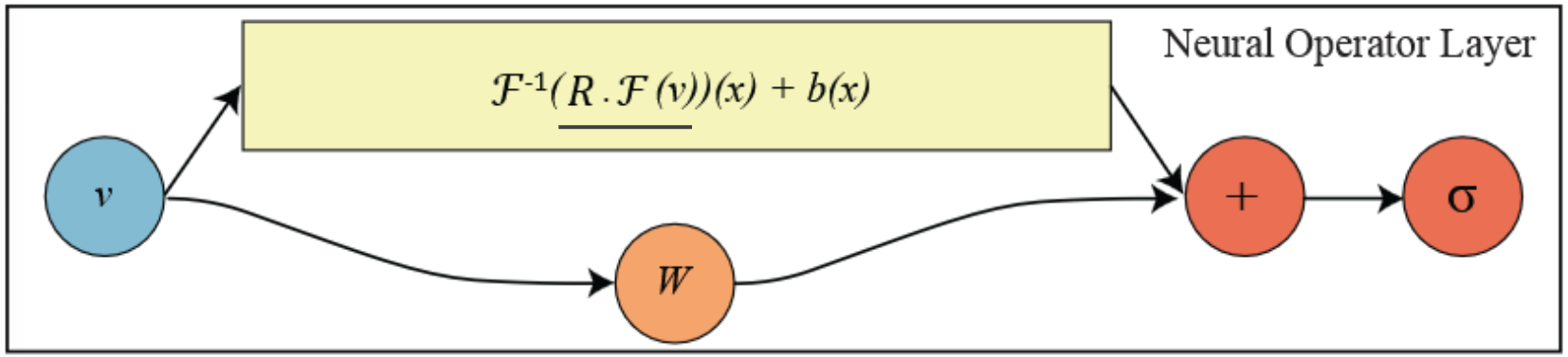


Frequency domain

Fourier Neural Operators

FNO Layer

Multiply learned transformations R by the input function in the frequency domain to model global, translation-invariant interactions across spatial dimensions. Then convert back



W is a local linear transform to retain local pointwise transformations in the spatial domain and compliment the operation in the frequency domain

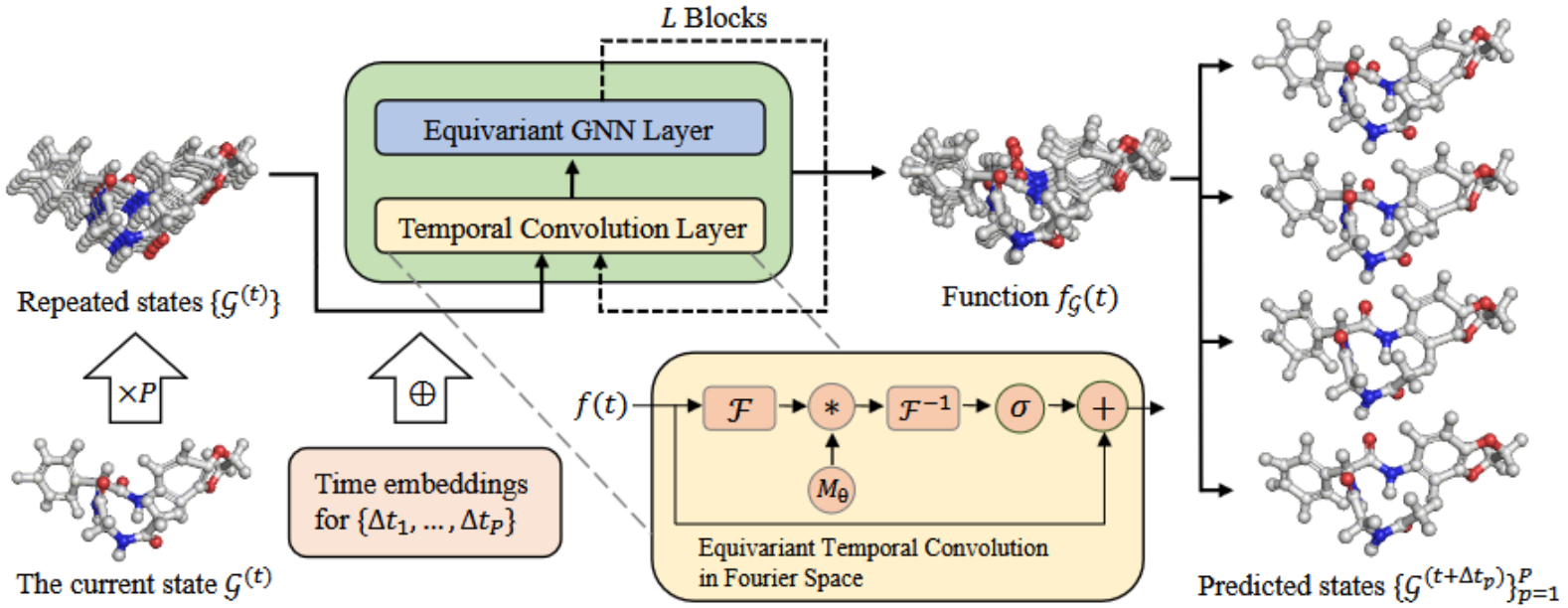
Summary

- We can use **graph neural networks** to make use of graph structure
- We can use several approaches to preserve salient translational/rotational transformations and SE(3)-**equivariance**
- We can use neural operators to more consistently respect **temporal correlations** and mitigate concerns about temporal discretization

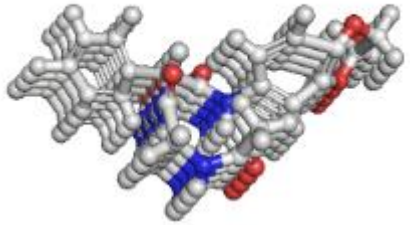
Equivariant Graph Neural Operator (EGNO)

Overview

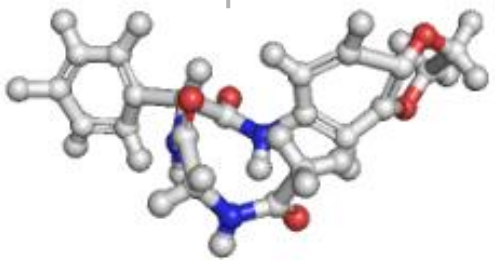
1. Use a modified FNOfor temporal modeling
2. Use an equivariant GNN layer for spatial modeling



IO & Training



Input trajectory function is trivially the current state repeated for different values of Δt: $\{G^{(t)}\}$

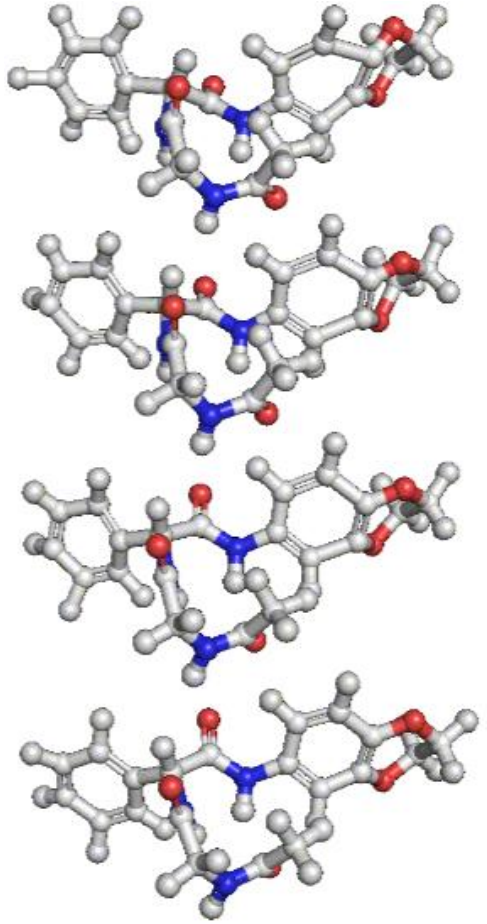


Current state: $G^{(t)}$

EGNO($G^{(t)}$)

Time embeddings: $t + \Delta t$

Function $f_G(t)$



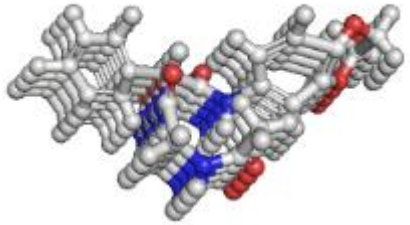
Predicted trajectory function: $\{G^{(t+\Delta t)}\}$

$$\min_{\theta} \mathbb{E}_{\mathcal{G}^{(t)} \sim p_{\text{data}}} \mathcal{L}(F_\theta(\mathcal{G}^{(t)})(t) - F^\dagger(\mathcal{G}^{(t)})(t)),$$

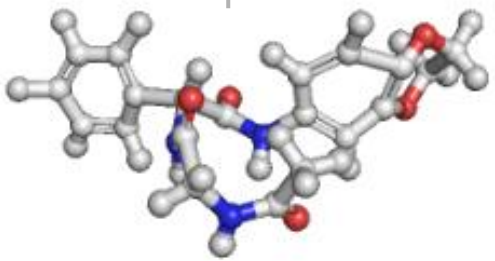
$$\min_{\theta} \mathbb{E}_{\mathcal{G}^{(t)} \sim p_{\text{data}}} \frac{1}{P} \sum_{p=1}^{P} ||F_\theta(\mathcal{G}^{(t)})(\Delta t_p) - F^\dagger(\mathcal{G}^{(t)})(\Delta t_p)||$$

Model — Ground truth

Basically just MAE



Input trajectory function is trivially the current state repeated for different values of Δt: $\{G^{(t)}\}$



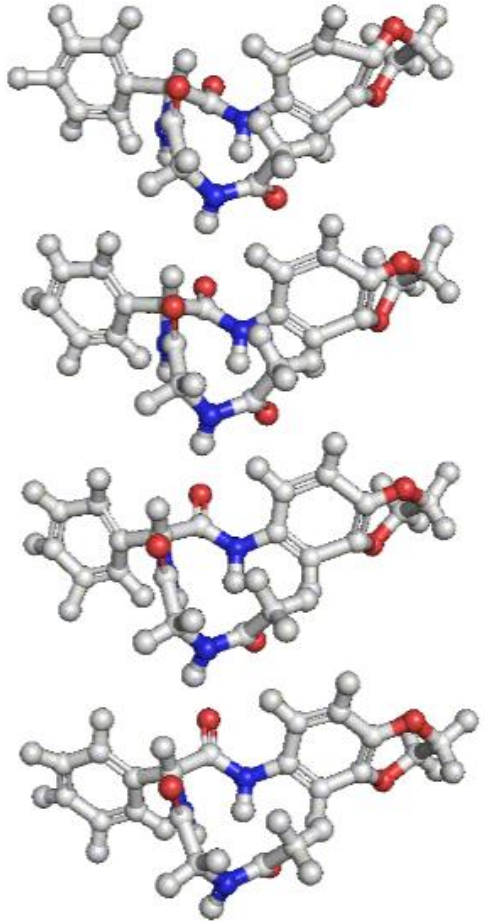
Current state: $G^{(t)}$

EGNO($G^{(t)}$)

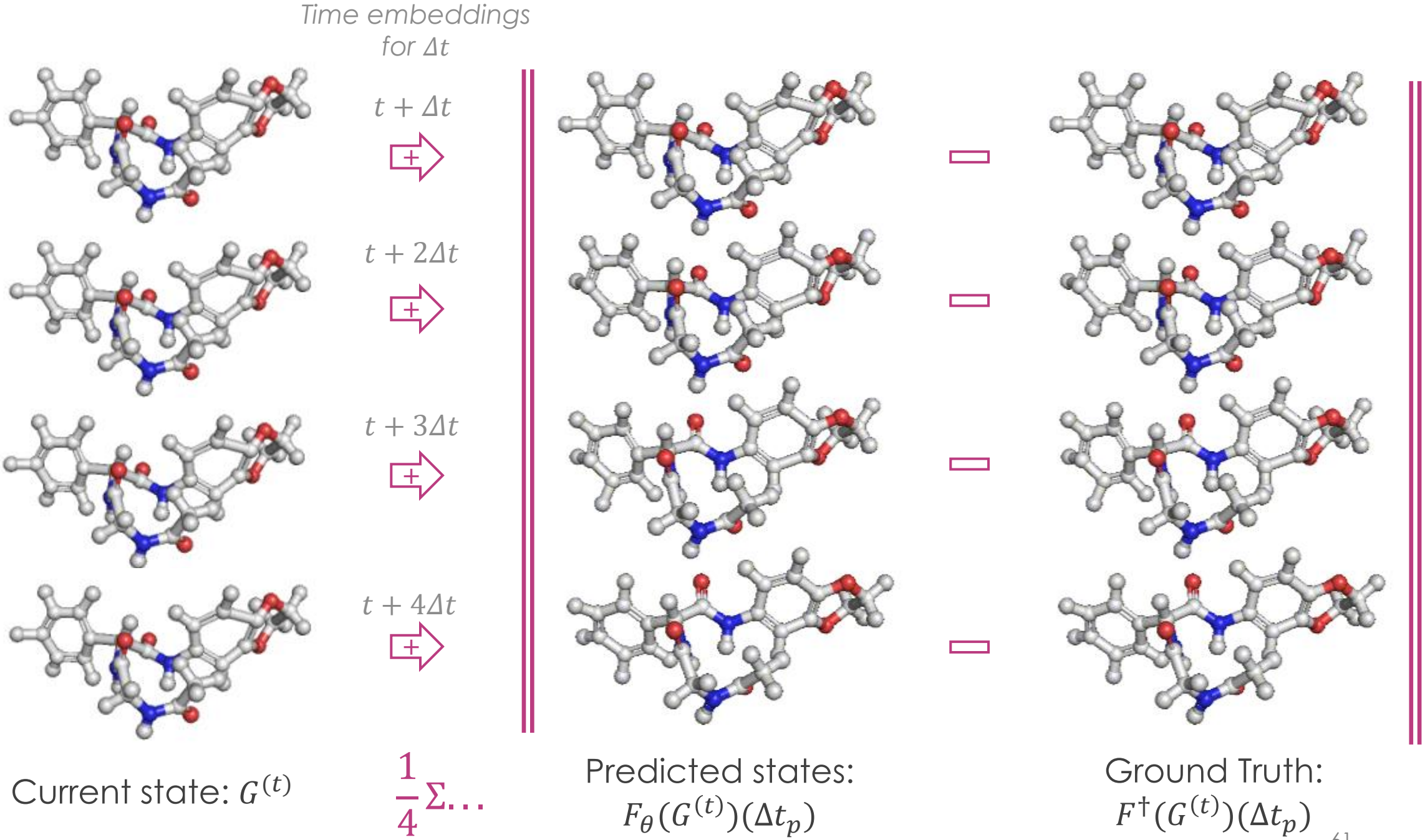
Time embeddings: $t + \Delta t$

Function $f_G(t)$

P: Number of timesteps
G: Graph as function of t
F: Prediction function

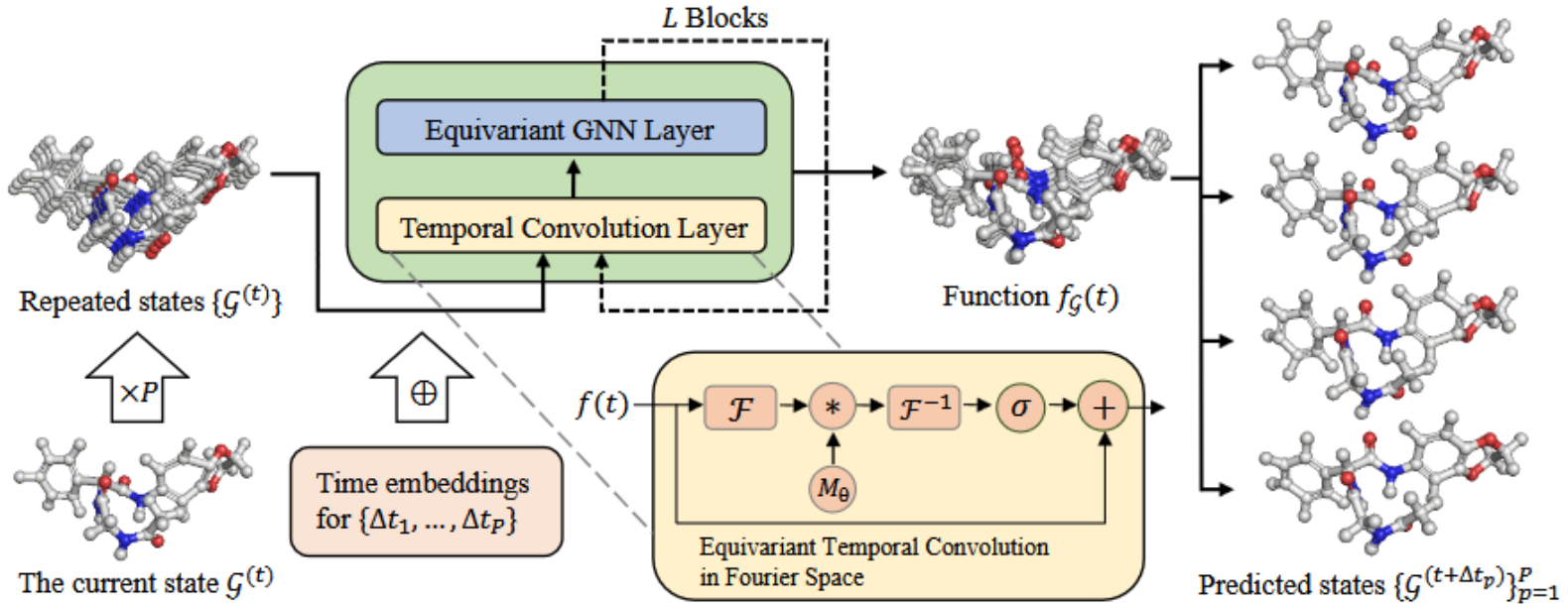


Predicted trajectory function: $\{G^{(t+\Delta t)}\}$


Time embeddings for Δt
$t + \Delta t$
$t + 2\Delta t$
$t + 3\Delta t$
$t + 4\Delta t$
Current state: $G^{(t)}$
$\frac{1}{4}\Sigma \ldots$
Predicted states: $F_\theta(G^{(t)})(\Delta t_p)$
Ground Truth: $F^\dagger(G^{(t)})(\Delta t_p)$

Overview

1. Use a modified FNO for temporal modeling
2. Use an equivariant GNN layer for spatial modeling



Equivariant Temporal Convolutional Layer

Unlike neural operators as originally prescribed, Xu et al. move nonlinearity and parameterization of the linear transform W to the integration layer K and leave W as a simple residual connection.

Neural operator

$$F_\theta := Q \circ \sigma(W_L + \mathcal{K}_L) \circ \cdots \circ \sigma(W_1 + \mathcal{K}_1) \circ P$$

Temporal convolution layer (2024)

$$(\mathcal{T}_\theta f)(t) = f(t) + \sigma((\mathcal{K}_\theta f)(t)),$$

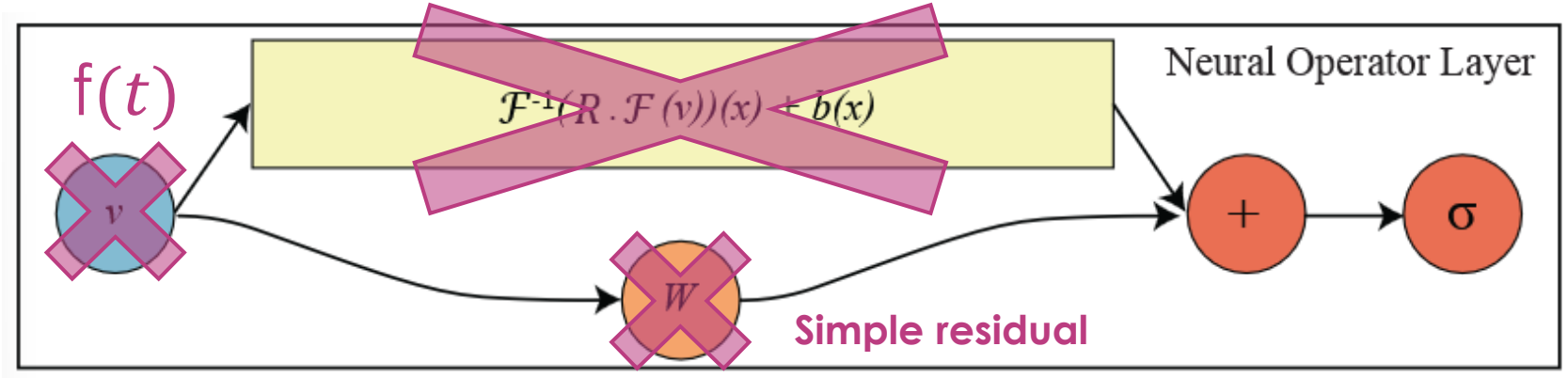
W: Linear transform
K: Integration layer
P: Lifting operator
Q: Projection operator
σ: Nonlinear function

Equivariant Temporal Convolutional Layer

FNO as originally prescribed also emphasize the **spatial domain** for things like discretized, fixed grids.

Xu et al. instead apply a convolution across only the **temporal domain** to model trajectories.

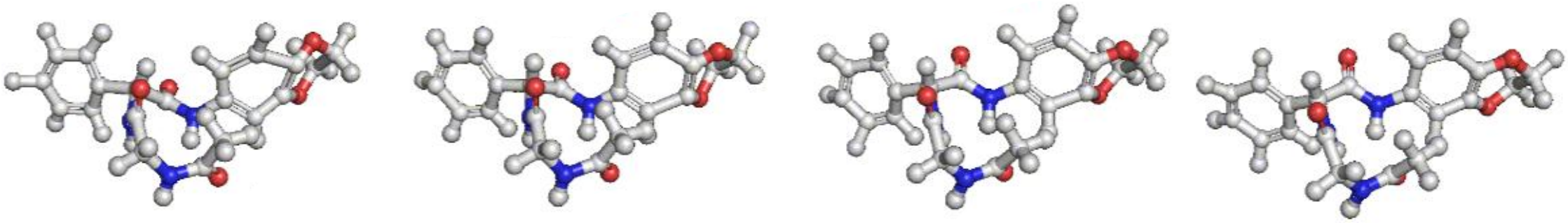
$$\mathcal{F}^{-1}\left(\begin{bmatrix}\mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0}\\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}}\end{bmatrix}\cdot\left(\mathcal{F}\begin{bmatrix}f_{\mathbf{h}}\\ f_{\mathbf{Z}}\end{bmatrix}\right)\right)(t),$$



Equivariant Temporal Convolutional Layer

Here, we aren't concerned about the frequency of change across the X, Y, and Z dimensions; we are concerned with how the frequency of change across timesteps.

Node positions and features move quickly? High frequency.
Node changes only a little? Low frequency.
Convolution is global across the trajectory, not space!



Time t

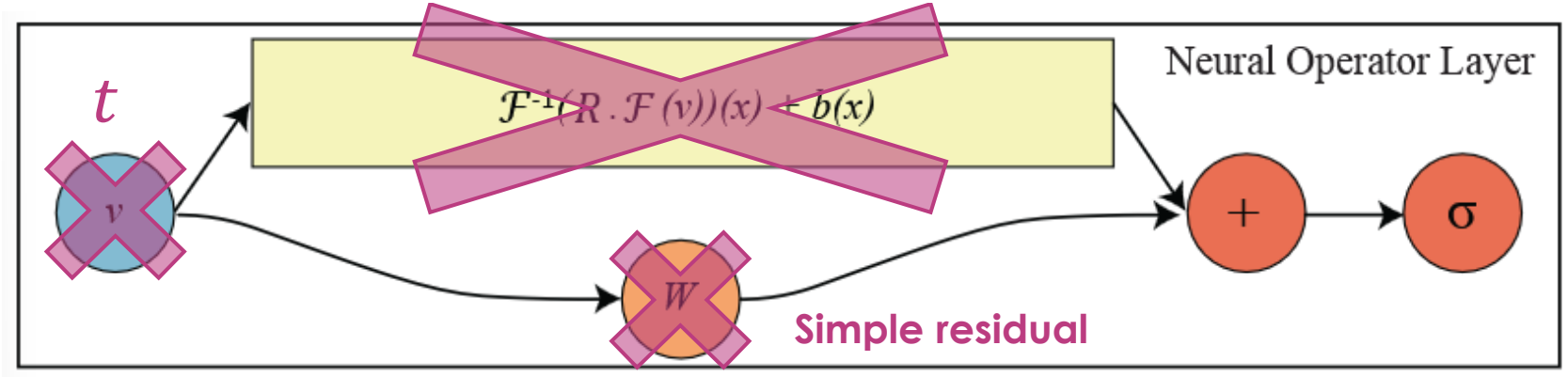
Equivariant Temporal Convolutional Layer

FNO as originally prescribed also emphasize the **spatial domain** for things like discretized, fixed grids.

Xu et al. instead apply a convolution across only the **temporal domain** to model trajectories.

$$\mathcal{F}^{-1}\left(\begin{bmatrix}\mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0}\\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}}\end{bmatrix}\cdot\left(\mathcal{F}\begin{bmatrix}f_{\mathbf{h}}\\ f_{\mathbf{Z}}\end{bmatrix}\right)\right)(t),$$

What about spatial equivariance?



Equivariant Temporal Convolutional Layer

1. Take current iteration of trajectory
2. Fourier transform along time & truncate to I freq. modes
3. Update scalar channels (i.e. $\boldsymbol{h}$)
4. Update vector channels (i.e. $\boldsymbol{Z}$)
5. Apply nonlinearity
6. Inverse Fourier transform

Equivariant Temporal Convolutional Layer

1. Take current iteration of trajectory
2. Fourier transform along time & truncate to I freq. modes
3. Update scalar channels (i.e. $\boldsymbol{h}$)
4. **Update vector channels (i.e. $\boldsymbol{Z}$)**
5. **Apply nonlinearity**
6. Inverse Fourier transform

FFT, IFFT, and scalar operations don't compromise equivariance, but vector operations can!

Equivariant Temporal Convolutional Layer

$$(\mathcal{K}_\theta f)(t) = \mathcal{F}^{-1}(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot (\mathcal{F}\begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix}))(t),$$

$$[\mathbf{M}_\theta^{\mathbf{h}} \cdot (\mathcal{F} f_{\mathbf{h}})]_{i,j} = \sum_{l=1}^{k} (\mathbf{M}_\theta^{\mathbf{h}})_{i,j,l} (\mathcal{F} f_{\mathbf{h}})_{i,l},$$

Scalars *Scalars*

$$[\mathbf{M}_\theta^{\mathbf{Z}} \cdot (\mathcal{F} f_{\mathbf{Z}})]_{i,s} = \sum_{l=1}^{m} (\mathbf{M}_\theta^{\mathbf{Z}})_{i,s,l} (\mathcal{F} f_{\mathbf{Z}})_{i,l},$$

Scalars ***3D vectors!***

To ensure equivariance, Xu et al…

- Mix scalar features $\boldsymbol{h}$ normally
  - No threat to equivariance
- Do **NOT** rotate vector features $\boldsymbol{Z}$
  - Only scalar multiplications, no component mixing
- Use only equivariant nonlinear activation functions $\hat{\sigma}$ for $\boldsymbol{Z}$
  - i.e. Scalar transforms that preserve directionality
- Shift CoM to zero then back for translational equivariance

$$\sigma(\mathcal{K}_\theta f) = [\sigma(\mathcal{K}_\theta f_{\mathbf{h}}), \hat{\sigma}(\mathcal{K}_\theta f_{\mathbf{Z}})]^{\mathbf{T}},$$

Equivariant Temporal Convolutional Layer

Proof of Theorem 4.1. Recall that our neural operator is given in full form by

$$\mathcal{T}_\theta f = f + \sigma \circ (\mathcal{F}^{-1}(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot (\mathcal{F} \begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix})) \tag{19}$$

The equivariance is shown directly

$$\mathbf{R} \cdot \mathcal{T}_\theta f = \mathbf{R} \cdot \left(f + \sigma \circ (\mathcal{F}^{-1}(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot (\mathcal{F} \begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix})) \right) \tag{20}$$

$$= \mathbf{R} \cdot f + \mathbf{R} \cdot \sigma \circ (\mathcal{F}^{-1}(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot (\mathcal{F} \begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix})) \tag{21}$$

$$= \mathbf{R} \cdot f + \sigma \circ \mathbf{R} \cdot (\mathcal{F}^{-1}(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot (\mathcal{F} \begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix})) \tag{22}$$

$$= \mathbf{R} \cdot f + \sigma \circ (\mathcal{F}^{-1}(\mathbf{R} \cdot \begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot (\mathcal{F} \begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix})) \tag{23}$$

$$= \mathbf{R} \cdot f + \sigma \circ (\mathcal{F}^{-1}(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot \mathbf{R} \cdot (\mathcal{F} \begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix})) \tag{24}$$

$$= \mathbf{R} \cdot f + \sigma \circ (\mathcal{F}^{-1}(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot (\mathcal{F}\mathbf{R} \cdot \begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix})) \tag{25}$$

$$= \mathcal{T}_\theta(\mathbf{R} \cdot f) \tag{26}$$

□

R: Rotation matrix
T: Neural operator
f: Function predicting $G^{(t)}$
$\boldsymbol{M}$: Learned linear transform
σ: Activation function

Equivariant GNN Layer

The authors claim that EGNO can stack any existing EGNN layers, but they test the original EGNN formulation by Satorras et al.

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$
$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)$$
$$\mathbf{m}_i = \sum_{j\neq i}\mathbf{m}_{ij}$$
$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$

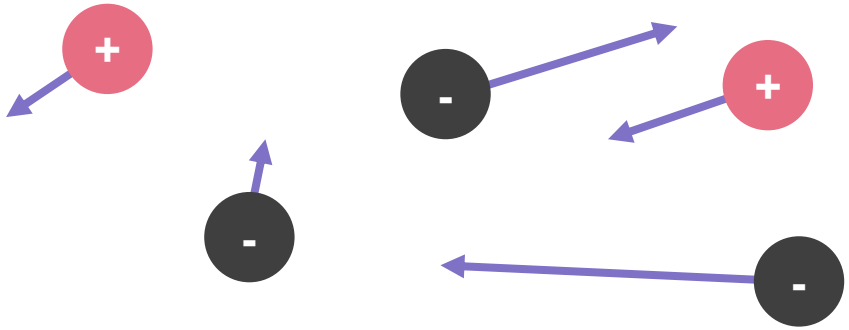
Equivariant graph convolutional layer
Satorras et al., 2021

Experiments & Results

1. N-body Simulation

- Following EGNN paper (Satorras et al., 2021)
- N charged particles in 3D space
- Each starts with position and velocity

With positive or negative charges, the particles either attract or repel each other.



1. N-body Simulation

Particles: $N = 5$
Time window: $\Delta T = 10$ timesteps
Time discretization/steps in ΔT: $P = 5$
EGNO-U: Uniform time discretization
EGNO-L: More samples near end of trajectory

Table 1. MSE in the N-body simulation.

	F-MSE
Linear (Satorras et al., 2021)	0.0819
SE(3)-Tr. (Fuchs et al., 2020)	0.0244
TFN (Thomas et al., 2018)	0.0155
MPNN (Gilmer et al., 2017)	0.0107
RF (Köhler et al., 2019)	0.0104
ClofNet (Du et al., 2022)	0.0065
EGNN (Satorras et al., 2021)	0.0071
EGNN-R	0.0720
EGNN-S	0.0070
EGNO	**0.0054**
	A-MSE
EGNN-R	0.0215
EGNN-S	0.0045
EGNO	**0.0022**

Table 2. F-MSE in N-body simulation *w.r.t.* different sizes of the training set.

|Train|	1000	3000	10000
EGNN	0.0094	0.0071	0.0051
EGNO-U ($P = 5$)	0.0082	0.0056	0.0036
EGNO-L ($P = 5$)	0.0071	0.0055	0.0038

Linear: Linear Dynamics
SE(3)-Tr.: SE(3)-Transformer
TFN: Tensor Field Networks
MPNN: Message Passing Neural Network
RF: Radial Field,
ClofNet: Complete Local Frames
EGNN: Equivariant Graph Neural Network
EGNN-R: Roll (iterative prediction)
EGNN-S: Sequential (layer outputs frames)

2. Motion Capture

- CMU Motion Capture dataset (CMU, 2003)
- They did not use Subject #60 (salsa dance)
  - They are cowards
- They used Subject #35 (walk) and Subject #9 (run)
  - They are cowards

2. Motion Capture

Time window: $\Delta T = 30$ timesteps
Time discretization/steps in ΔT: $P = 5$

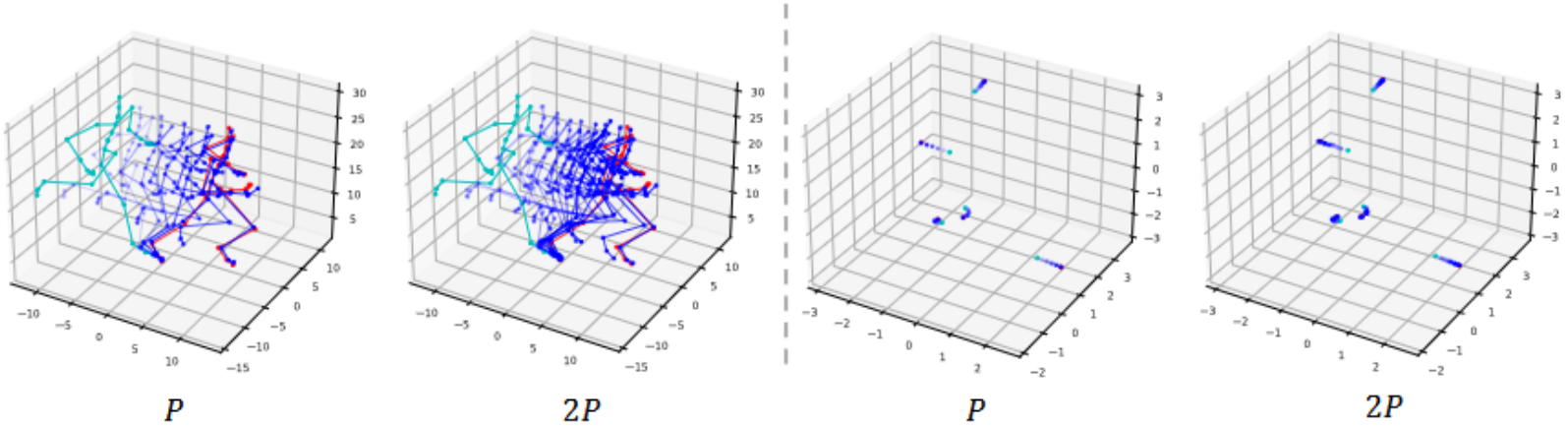


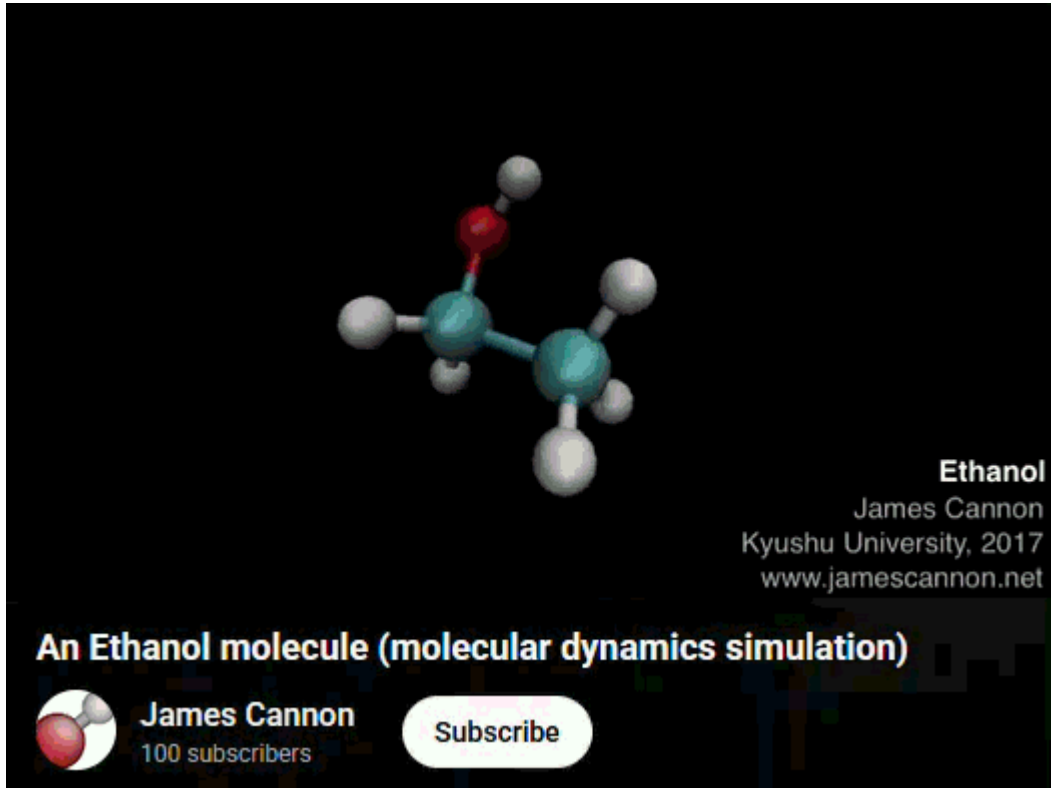
Figure 3. Qualitative results of zero-shot generalization towards discretization steps. Sub-figures indexed by P or $2P$ have P or $2P$ timesteps while sharing exactly the same initial conditions. Left: Motion Capture `Run`. Right: N-body simulation. Best viewed in color.

Table 4. MSE ($\times 10^{-2}$) on the motion capture dataset. The upper part is F-MSE for `S2S` and the lower part is A-MSE for `S2T`.

	Subject #35 `Walk`	Subject #9 `Run`
MPNN	36.1 ±1.5	66.4 ±2.2
RF	188.0 ±1.9	521.3±2.3
TFN	32.0 ±1.8	56.6 ±1.7
SE(3)-Tr.	31.5 ±2.1	61.2 ±2.3
EGNN	28.7 ±1.6	50.9 ±0.9
EGNN-R	90.7 ±2.4	816.7 ±2.7
EGNN-S	26.4 ±1.5	54.2 ±1.9
EGNO	**8.1 ±1.6**	**33.9 ±1.7**
EGNN-R	32.0 ±1.6	277.3 ±1.8
EGNN-S	14.3 ±1.2	28.5 ±1.3
EGNO	**3.5 ±0.5**	**14.9 ±0.9**

3. Molecular Dynamics: Small Molecules

- MD17 dataset
  - Chmiela et al., 2017
- Describes trajectories of 8 small molecules



≈1 picosecond of small molecule dynamics

3. Molecular Dynamics: Small Molecules

Table 5. MSE ($\times 10^{-2}$) on MD17 dataset. Upper part: F-MSE for `S2S`. Lower part: A-MSE for `S2T`.

	Aspirin	Benzene	Ethanol	Malonaldehyde	Naphthalene	Salicylic	Toluene	Uracil
RF (Köhler et al., 2019)	10.94±0.01	103.72±1.29	4.64±0.01	13.93±0.03	0.50±0.01	1.23±0.01	10.93±0.04	0.64±0.01
TFN (Thomas et al., 2018)	12.37±0.18	58.48±1.98	4.81±0.04	13.62±0.08	0.49±0.01	1.03±0.02	10.89±0.01	0.84±0.02
SE(3)-Tr. (Fuchs et al., 2020)	11.12±0.06	68.11±0.67	4.74±0.13	13.89±0.02	0.52±0.01	1.13±0.02	10.88±0.06	0.79±0.02
EGNN (Satorras et al., 2021)	14.41±0.15	62.40±0.53	4.64±0.01	13.64±0.01	0.47±0.02	1.02±0.02	11.78±0.07	0.64±0.01
EGNN-R (Satorras et al., 2021)	14.51±0.19	62.61±0.75	4.94±0.21	17.25±0.05	0.82±0.02	1.35±0.02	11.59±0.04	1.11±0.02
EGNN-S (Satorras et al., 2021)	9.50±0.10	66.45±0.89	4.63±0.01	12.88±0.01	0.45±0.01	1.00±0.02	10.78±0.05	0.60±0.01
EGNO	**9.18**±0.06	**48.85**±0.55	**4.62**±0.01	**12.80**±0.02	**0.37**±0.01	**0.86**±0.02	**10.21**±0.05	**0.52**±0.02
EGNN-R (Satorras et al., 2021)	12.07±0.11	23.73±0.30	3.44±0.17	13.38±0.03	0.63±0.01	1.15±0.02	5.04±0.02	0.89±0.01
EGNN-S (Satorras et al., 2021)	9.49±0.12	29.99±0.65	3.29±0.01	11.21±0.01	0.43±0.01	1.36±0.02	4.85±0.04	0.68±0.01
EGNO	**7.37**±0.07	**22.41**±0.31	**3.28**±0.02	**10.67**±0.01	**0.32**±0.01	**0.77**±0.01	**4.58**±0.03	**0.47**±0.01

Time discretization/steps: $P = 8$

3. Molecular Dynamics: Proteins

- Adk equilibrium trajectory dataset
  - Seyler & Beckstein, 2017
- Describes the trajectory of apo-adenylate kinase
  - Enzyme that regulates energy in the cells
- Unlike small molecule sim, models large structural changes

Apo-Adenylate Kinase, Halder et al., 2017

3. Molecular Dynamics: Proteins

Table 6. F-MSE on AdK equilibrium trajectory dataset.

Linear	RF	MPNN	EGNN	EGHN	EGNO	EGHNO
2.890	2.846	2.322	2.735	2.034	2.231	**1.801**

EGHN: Equivariant Graph Hierarchy-Based Neural Networks (Han et al., 2022)
EGHNO: EGHN + EGNO

Time discretization/steps: $P = 4$

4. Ablation Studies

To test the usefulness of each element of EGNO, 4 variants are tested on the N-body system and motion capture tasks

- I: Complete, vanilla EGNO
- II: Only features $\boldsymbol{h}$ and $\boldsymbol{x}$, but no $\boldsymbol{v}$ in temporal convolution
- III: Only features $\boldsymbol{h}$ in temporal convolution
- IV: EGNN with time embeddings only

Also, multiple number of modes I (ie. dimensionality in frequency domain) are tested.

4. Ablation Studies

		h	x	v	N-body	Mocap-Run
EGNO	I	✓	✓	✓	**0.0055**	**33.9**
	II	✓	✓	-	0.0057	35.6
	III	✓	-	-	0.0061	39.1
	IV	-	-	-	0.0072	48.2
EGNN					0.0071	50.9

I: Complete, vanilla EGNO
II: Only features $\boldsymbol{h}$ and $\boldsymbol{x}$, but no $\boldsymbol{v}$ in temporal convolution
III: Only features $\boldsymbol{h}$ in temporal convolution
IV: EGNN with time embeddings only

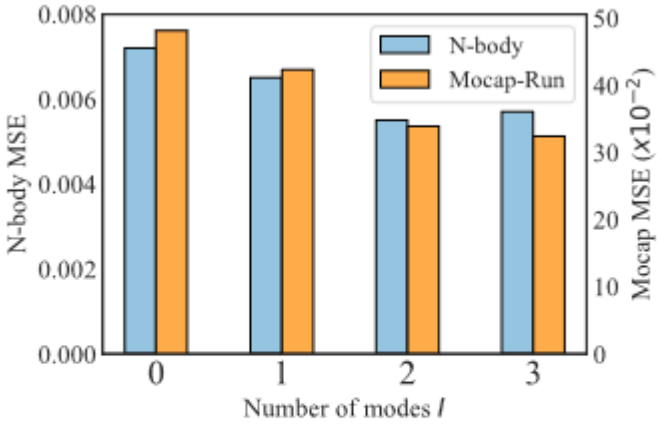


Figure 2. Ablation studies on the number of modes I on N-body simulation and Mocap-Run datasets.

Authors suggest high values of I lead to redundant frequencies with noisy patterns and consequential overfitting

5. Discretization Generalization

Xu et al. tested zero-shot generalization to different timesteps.

In other words, they tried different values of P and showed surprising stability despite coarser-grained training data.

Table 3. F-MSE in N-body simulation *w.r.t.* different numbers of time steps P.

|Train|=3000	EGNO-U	EGNO-L
$P = 2$	0.0062	0.0067
$P = 5$	0.0056	0.0055
$P = 10$	0.0057	0.0054

Performance was as stable for P=5 as it was with P=10, indicating generalization despite coarser-grained training

The authors attribute this to stability granted by convolution across the temporal frequency domain

Thank You

https://www.youtube.com/watch?v=DqOz_LhzXoo

https://www.youtube.com/watch?v=Ee7xbSBFWx4

References (1)

Now with inconsistent formatting!

- M. Xu *et al.*, 'Equivariant Graph Neural Operator for Modeling 3D Dynamics', *arXiv [cs.LG]*. 2024. https://arxiv.org/abs/2401.11037
- Nfusion, 'Barnes-Hut method Nbody simulation (N=2M)', YouTube, https://www.youtube.com/watch?v=81FjTTB1mtc
- Lars Nyland, Mark Harris, Jans Prins. GPU Gems 3. 'Chapter 31. Fast N-Body Simulation with CUDA'. NVIDIA. https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda
- Illustris Project, https://www.illustris-project.org/about/
- Takahiro Harada. GPU Gems 3. 'Chapter 29.Real-Time Rigid Body Simulation on GPUs'. NVIDIA. https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-29-real-time-rigid-body-simulation-gpus
- Tuxedo Labs, 'Teardown Game', https://www.youtube.com/watch?v=Ov5GxFlHqUQ
- Matthias Müller, 'Soft Body Simulation'. https://matthias-research.github.io/pages/tenMinutePhysics/10-softBodies.html
- A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, 'Learning to Simulate Complex Physics with Graph Networks', *arXiv [cs.LG]*. 2020. https://arxiv.org/abs/2002.09405
- W. Wu, Z. Qi, and L. Fuxin, 'PointConv: Deep Convolutional Networks on 3D Point Clouds', *arXiv [cs.CV]*. 2020. https://arxiv.org/abs/1811.07246
- N. Thomas *et al.*, 'Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds', *arXiv [cs.LG]*. 2018. https://arxiv.org/abs/1802.08219

References (2)

- F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling, 'SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks', arXiv [cs.LG]. 2020. https://arxiv.org/abs/2006.10503
- V. G. Satorras, E. Hoogeboom, and M. Welling, 'E(n) Equivariant Graph Neural Networks', *arXiv [cs.LG]*. 2022. https://arxiv.org/abs/2102.09844
- Y. Liu, L. Wang, M. Liu, X. Zhang, B. Oztekin, and S. Ji, 'Spherical Message Passing for 3D Graph Networks', *arXiv [cs.LG]*. 2022. https://arxiv.org/abs/2102.05013
- J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, 'Neural Message Passing for Quantum Chemistry', *arXiv [cs.LG]*. 2017. https://arxiv.org/abs/1704.01212
- N. Kovachki *et al.*, 'Neural operator: learning maps between function spaces with applications to PDEs', *arXiv [cs.LG]*. 2021. https://arxiv.org/abs/2108.08481
- CMU. Carnegie-mellon motion capture database. 2003. https://mocap.cs.cmu.edu/
- S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, 'Machine learning of accurate energy-conserving molecular force fields', *Science Advances*, vol. 3, no. 5, May 2017. http://dx.doi.org/10.1126/sciadv.1603015
- S. Seyler and O. Beckstein, 'Molecular dynamics trajectory for benchmarking MDAnalysis', 6 2017. https://figshare.com/articles/dataset/Molecular_dynamics_trajectory_for_benchmarking_MDAnalysis/5108170
- R. Halder, R. N. Manna, S. Chakraborty, and B. Jana, 'Modulation of the Conformational Dynamics of Apo-Adenylate Kinase through a π–Cation Interaction', *J. Phys. Chem. B*, vol. 121, no. 23, pp. 5699–5708, June 2017. https://pubs.acs.org/doi/10.1021/acs.jpcb.7b01736
- J. Han, W. Huang, T. Xu, and Y. Rong, 'Equivariant Graph Hierarchy-Based Neural Networks', *arXiv [cs.LG]*. 2022. https://arxiv.org/abs/2202.10643