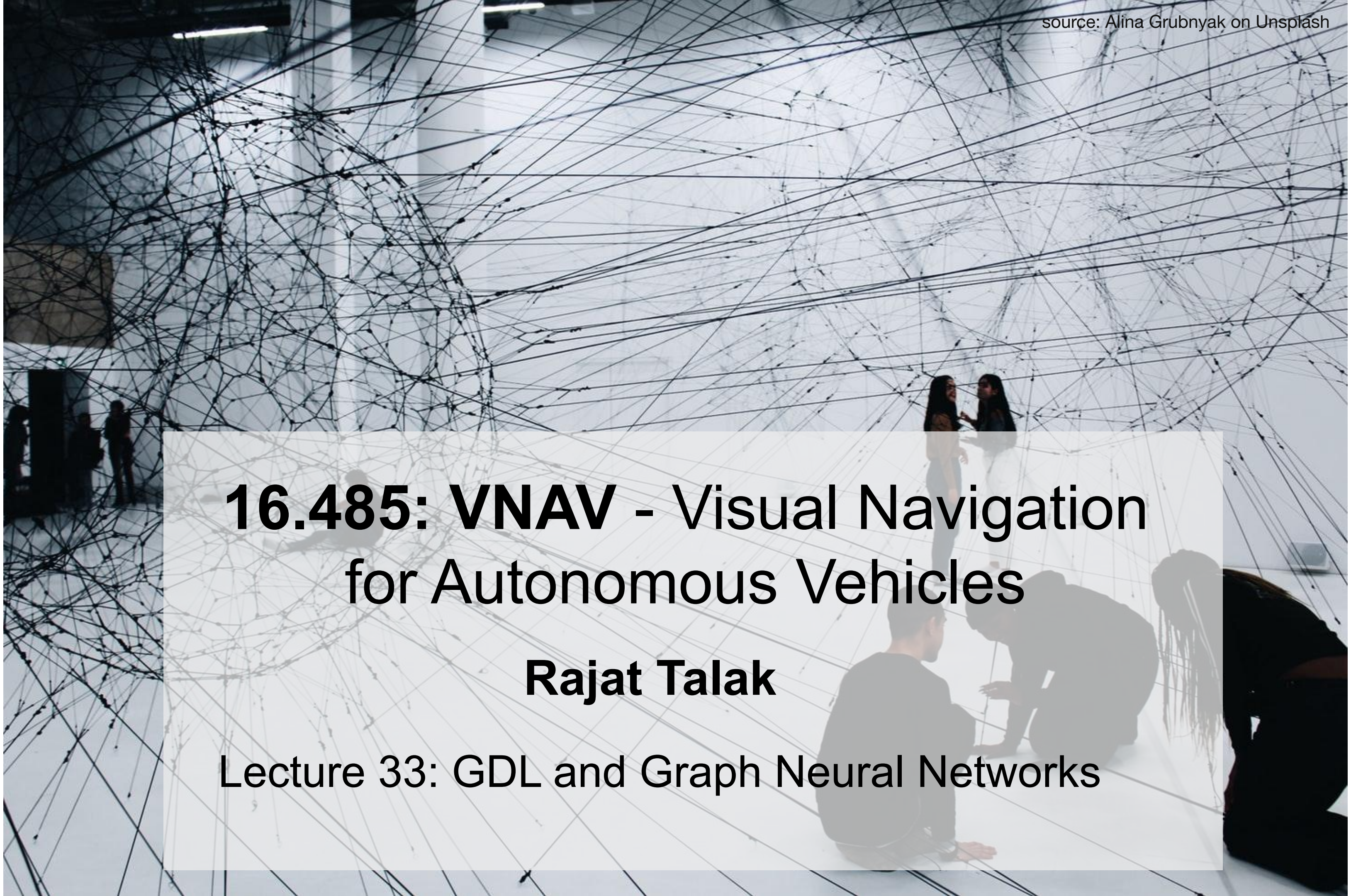


source: Alina Grubnyak on Unsplash

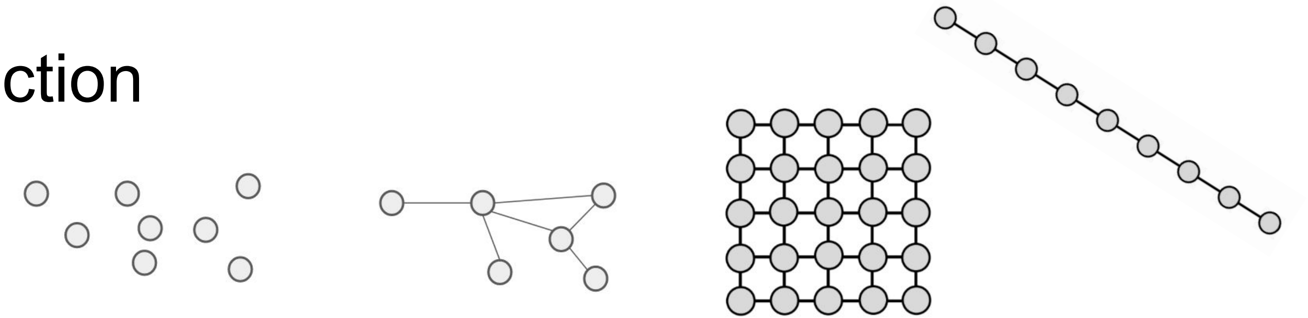
16.485: VNAV - Visual Navigation for Autonomous Vehicles

Rajat Talak

Lecture 33: GDL and Graph Neural Networks

Recap: Abstraction

Ω



$$\mathcal{X}(\Omega) = \{x : \Omega \to \mathbb{R}\}$$

$$\mathbb{S} = G$$

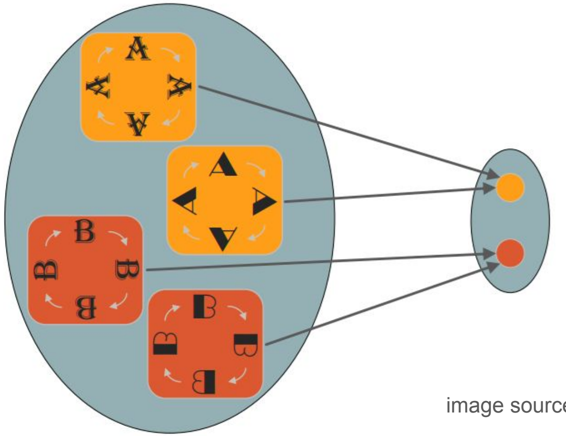
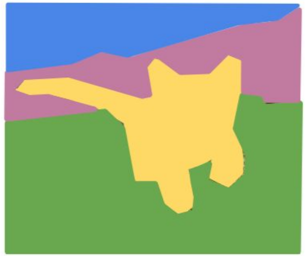


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Recap: Abstraction

Classification

G-invariant

$$\mathcal{F}_C = \{f : \mathcal{X}(\Omega) \to \mathbb{R} \mid f(g \cdot x) = f(x) \, \forall \, g \in G\}$$

Segmentation

G-equivariant

$$\mathcal{F}_S = \{f : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega) \mid f(g \cdot x) = g \cdot f(x) \, \forall \, g \in G\}$$

Today

- Geometric Deep Learning Blueprint
- Apply the Blueprint to Sets
  - PointNet, Transformers
- Apply the Blueprint to Graphs
  - Graph Neural Networks
  - Scene Graphs
  - Expressivity Limits of Graph Neural Networks

Translation Equivariance and Convolution

$$\Omega = [d] \times [d]$$

$$G = \{S_{k,l} \mid S_{k,l} = \text{shift by } (k, l)\}$$

$$\mathcal{X}(\Omega) = \mathbb{R}^{d \times d}$$

$$(S_{k,l} \cdot x)\,(i, j) = x(i \oplus k, j \oplus l)$$

$L : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega)$ is a linear map

Translation Equivariance and Convolution

$$\Omega = [d] \times [d]$$

$$G = \{S_{k,l} \mid S_{k,l} = \text{shift by } (k, l)\}$$

$$\mathcal{X}(\Omega) = \mathbb{R}^{d \times d}$$

$$(S_{k,l} \cdot x)\,(i, j) = x(i \oplus k, j \oplus l)$$

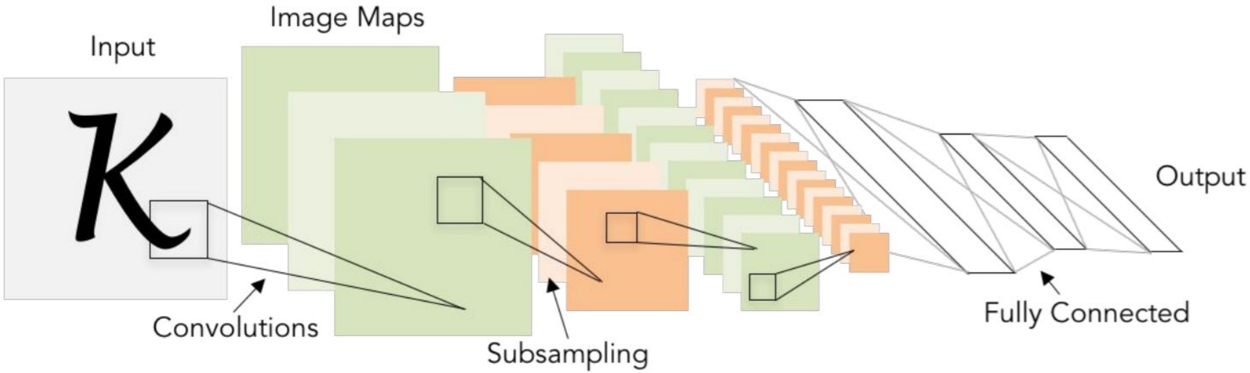
$L : \mathcal{X}(\Omega) \to \mathcal{X}(\Omega)$ is a linear map

Theorem

$f(x) = \sigma(L(x))$ is G-equivariant

$L(x)$ is a convolution

What does this mean?



Theorem

$$f^{\text{inv}} : \mathcal{X} \to \mathbb{R}$$

$$f_1^{\text{eq}}, f_2^{\text{eq}}, \ldots f_L^{\text{eq}} : \mathcal{X} \to \mathcal{X}$$

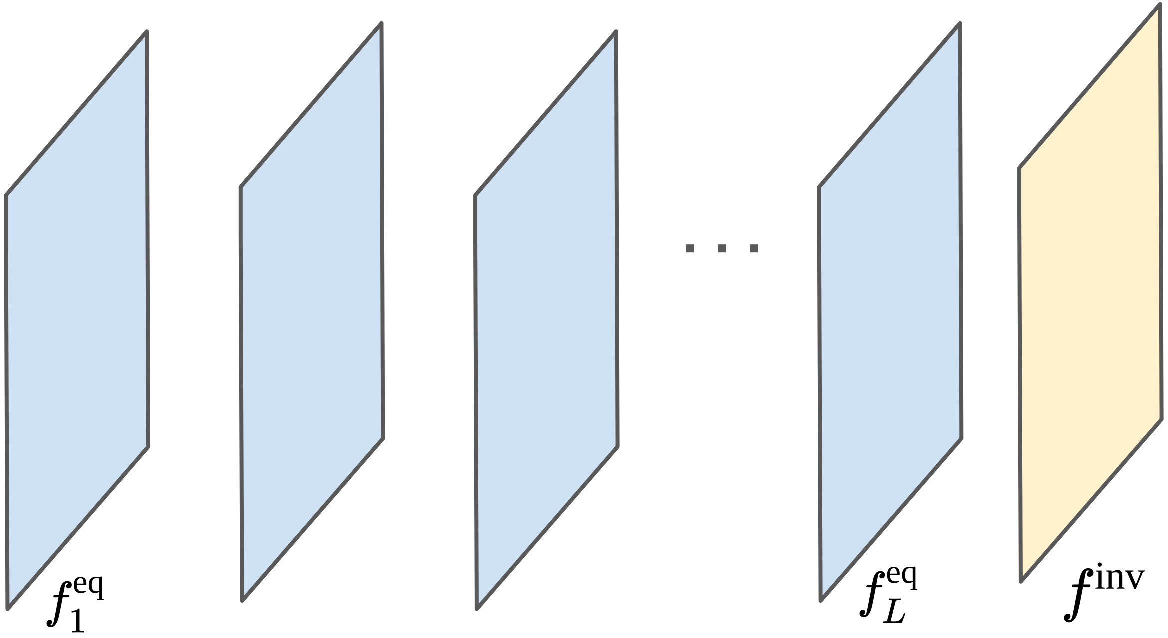
$$\Rightarrow$$

$$f^{\text{inv}} \cdot f_L^{\text{eq}} \cdot f_{L-1}^{\text{eq}} \cdots f_1^{\text{eq}} : \mathcal{X} \to \mathbb{R}$$

is G-invariant

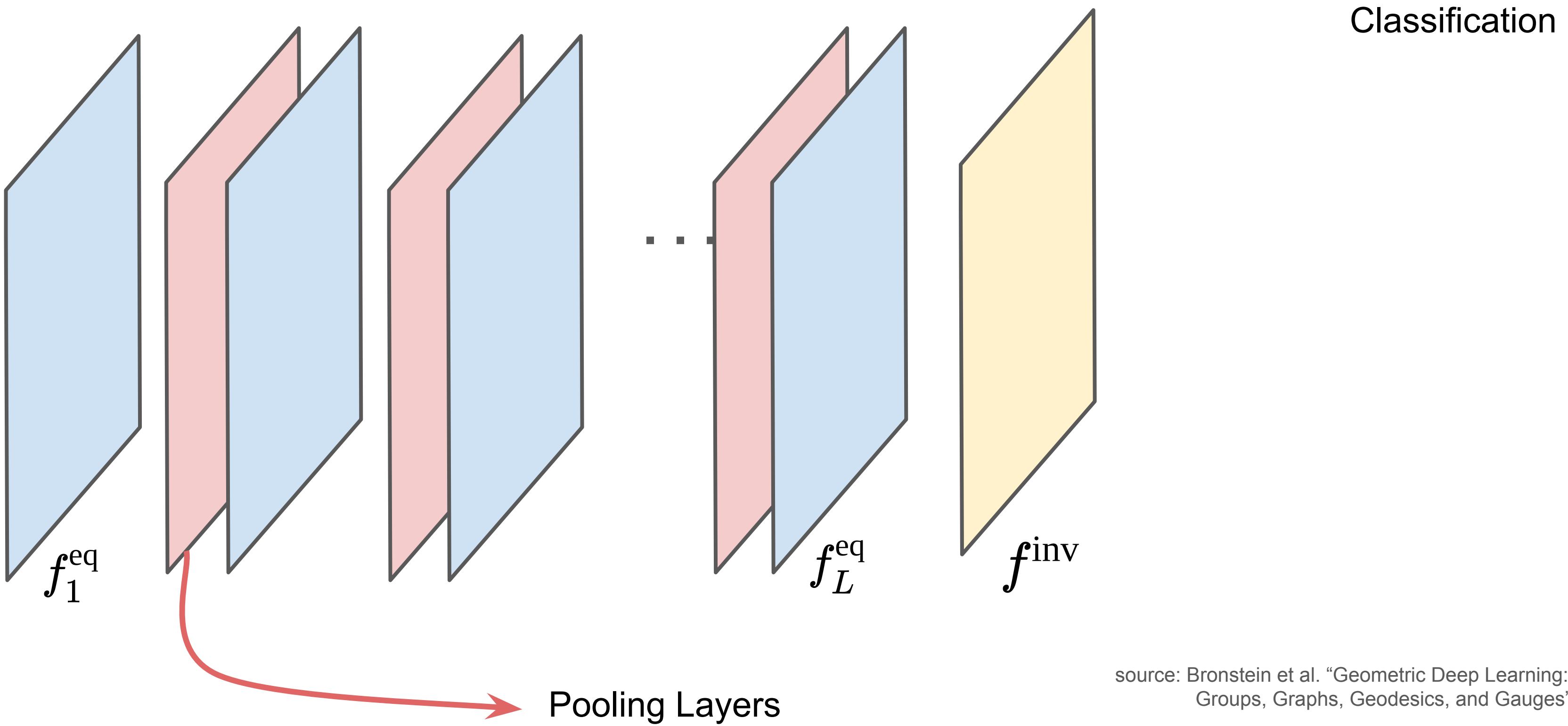
Geometric Deep Learning Blueprint

Classification



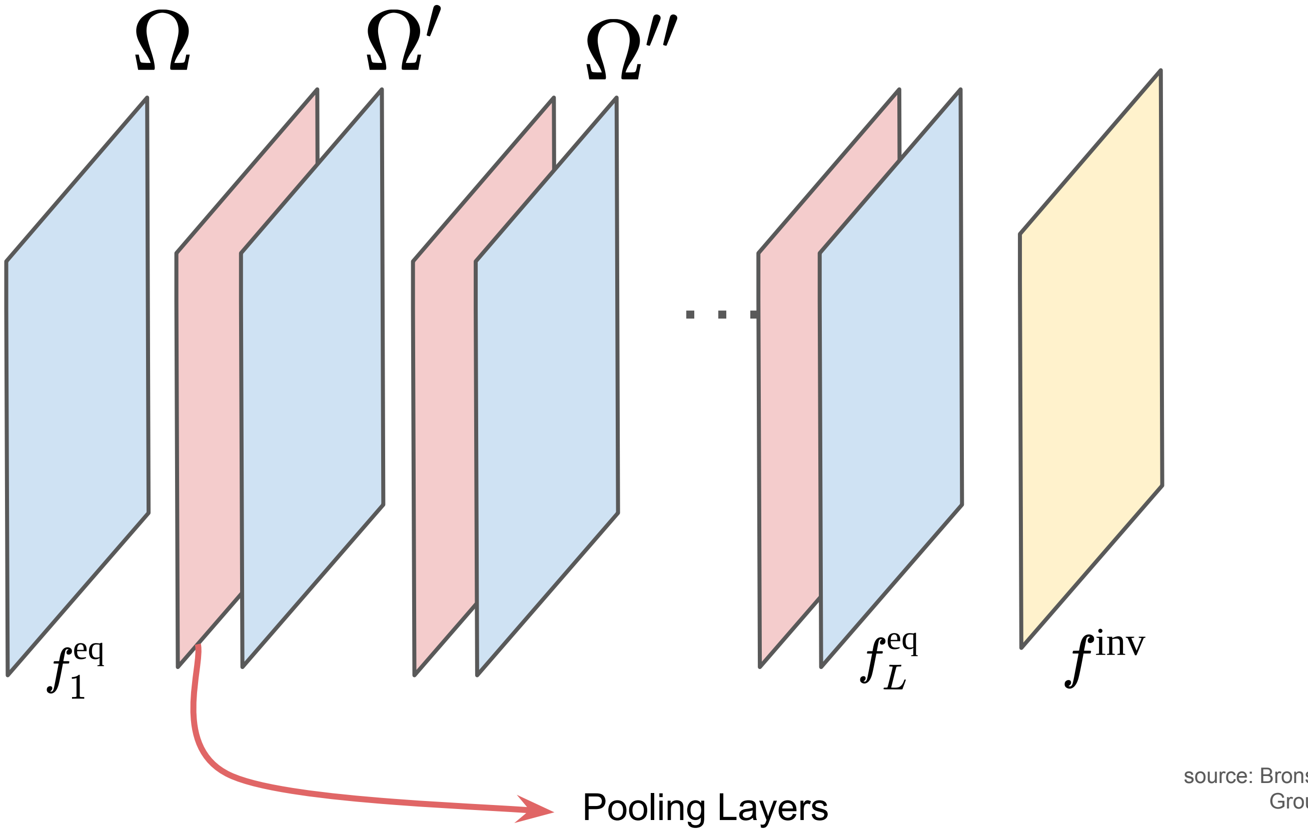
source: Bronstein et al. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges” 2021.

Geometric Deep Learning Blueprint



source: Bronstein et al. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges” 2021.

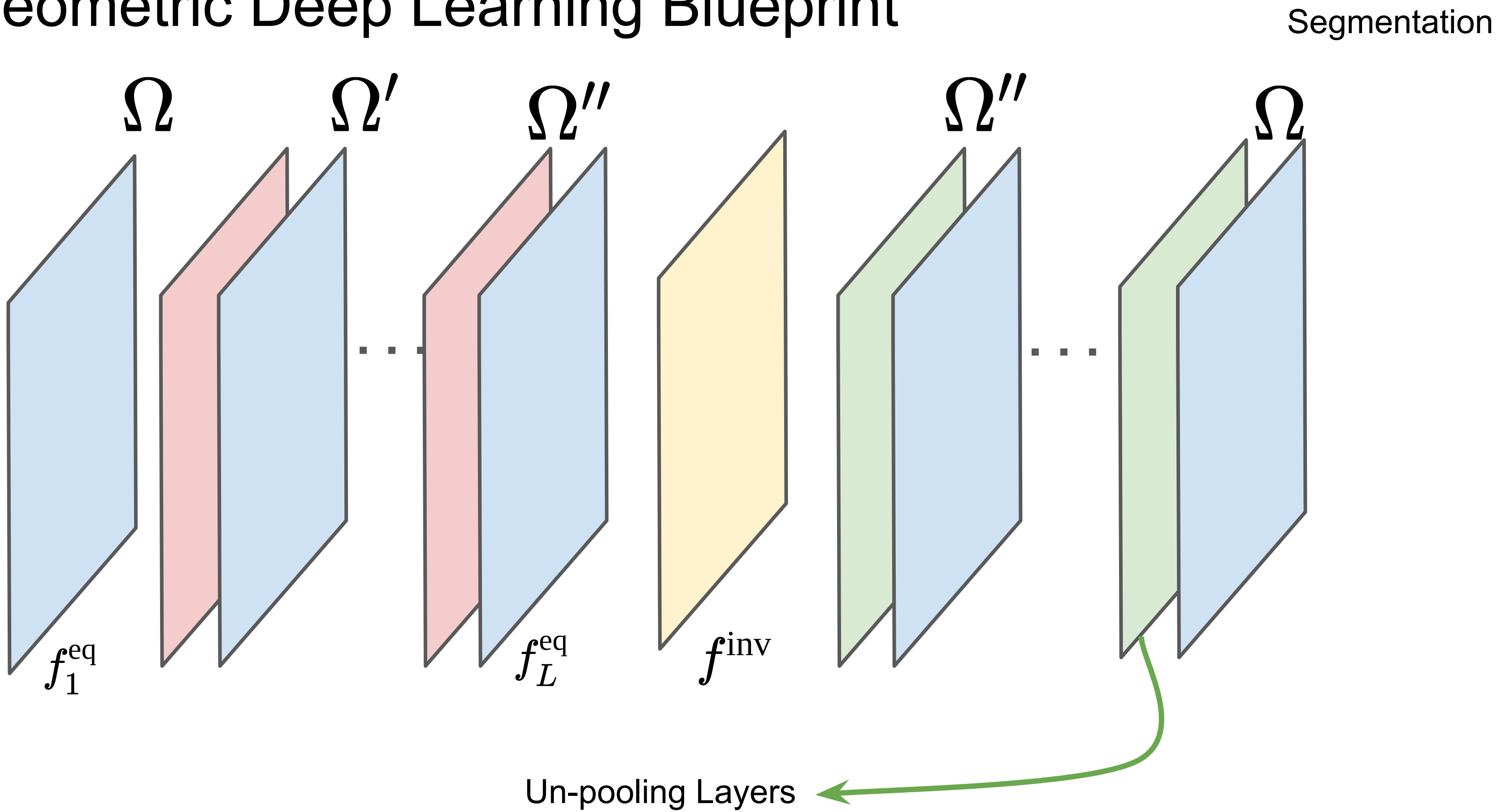
Geometric Deep Learning Blueprint



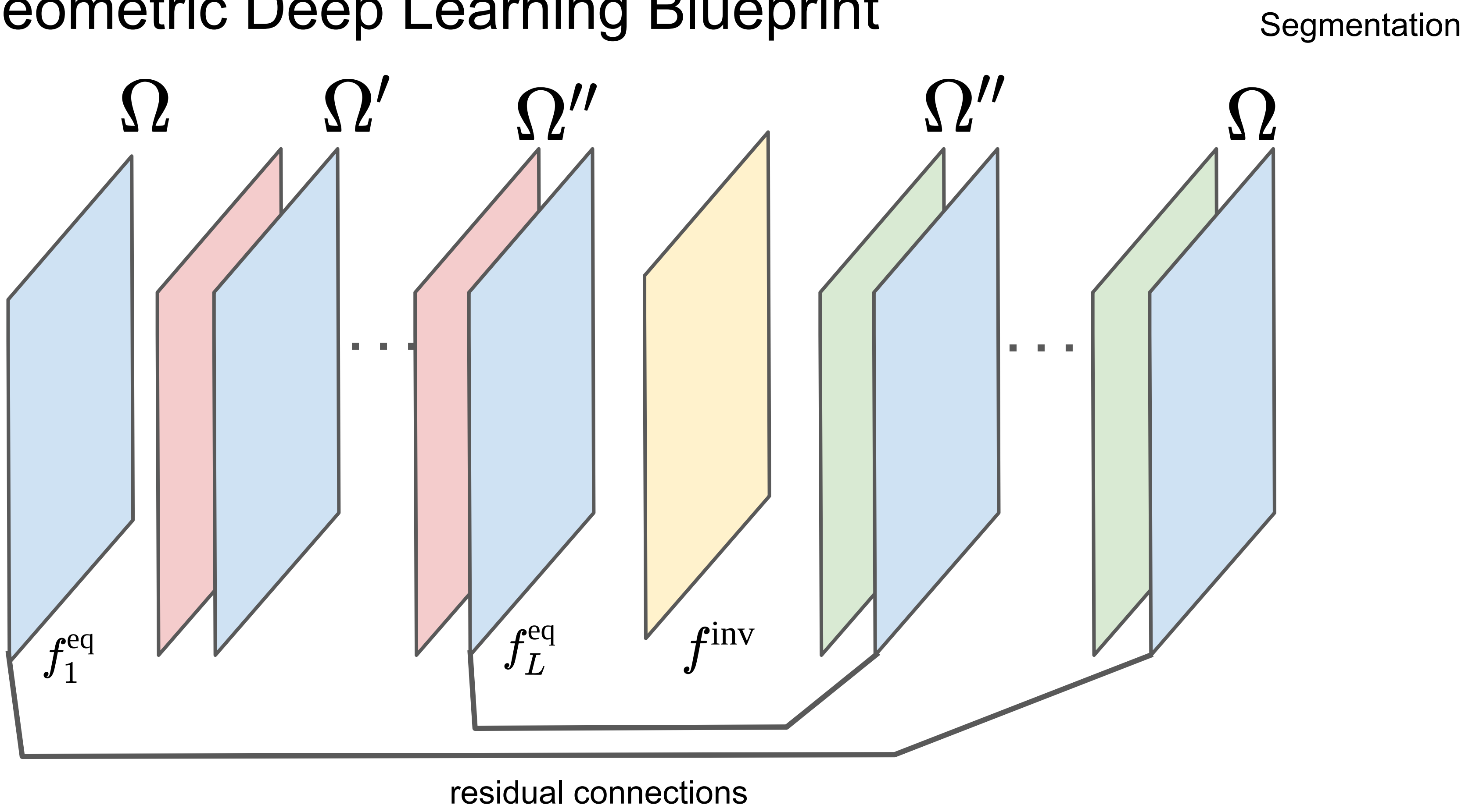
Classification

source: Bronstein et al. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges" 2021.

Geometric Deep Learning Blueprint



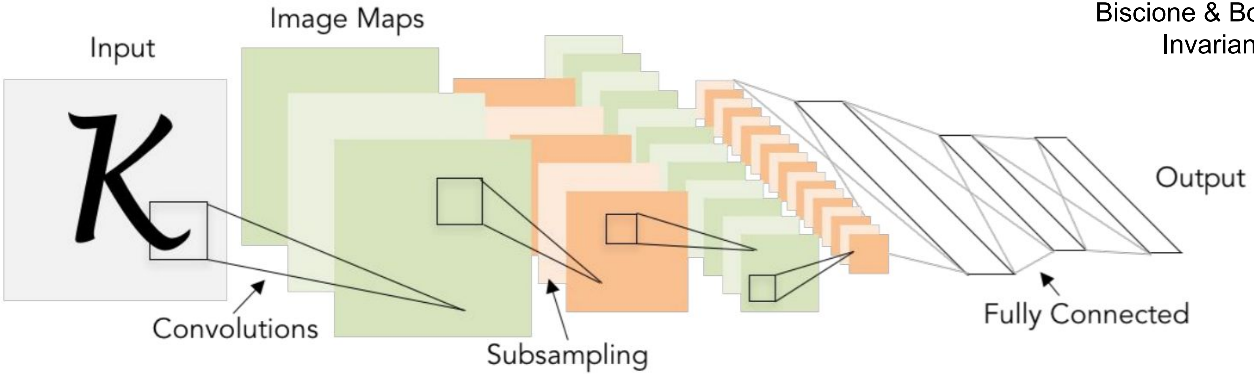
Geometric Deep Learning Blueprint



A small twist to our tale …

CNNs are not strictly translation invariant!

Biscione & Bowers "Learning Translation Invariance in CNNs" NeurIPS 2020



Theorem

$$f^{\text{inv}} : \mathcal{X} \to \mathbb{R}$$

$$f_1^{\text{eq}}, f_2^{\text{eq}}, \dots f_L^{\text{eq}} : \mathcal{X} \to \mathcal{X}$$

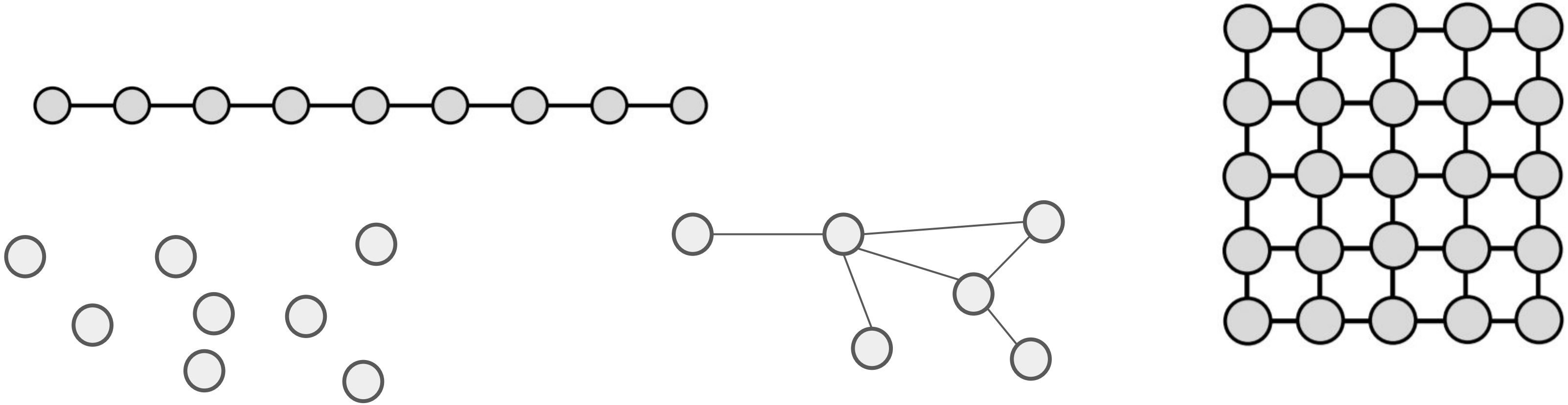
⇒

$$f^{\text{inv}} \cdot f_L^{\text{eq}} \cdot f_{L-1}^{\text{eq}} \cdots f_1^{\text{eq}} : \mathcal{X} \to \mathbb{R}$$

is G-invariant

Using the Blueprint

Suffices to find invariant and equivariant functions on different domains



Sets

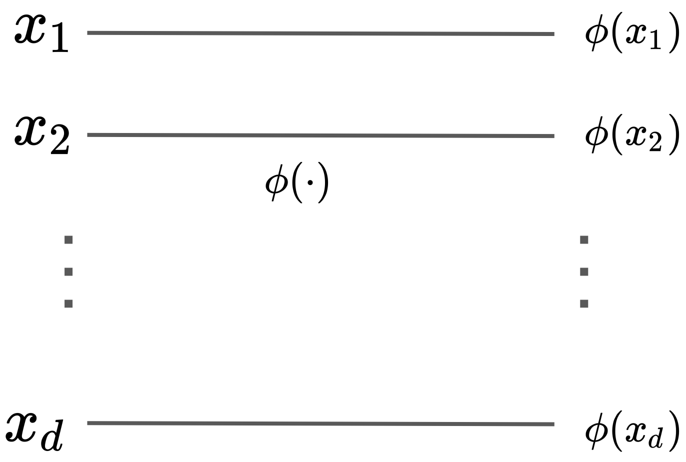
Equivariance over Sets

$\Omega = [d]$ $\mathcal{X}(\Omega) = \mathbb{R}^d$ $G = \{P \mid P = d \times d \text{ permutation matrix}\}$

$$f(P \cdot x) = P \cdot f(x)$$

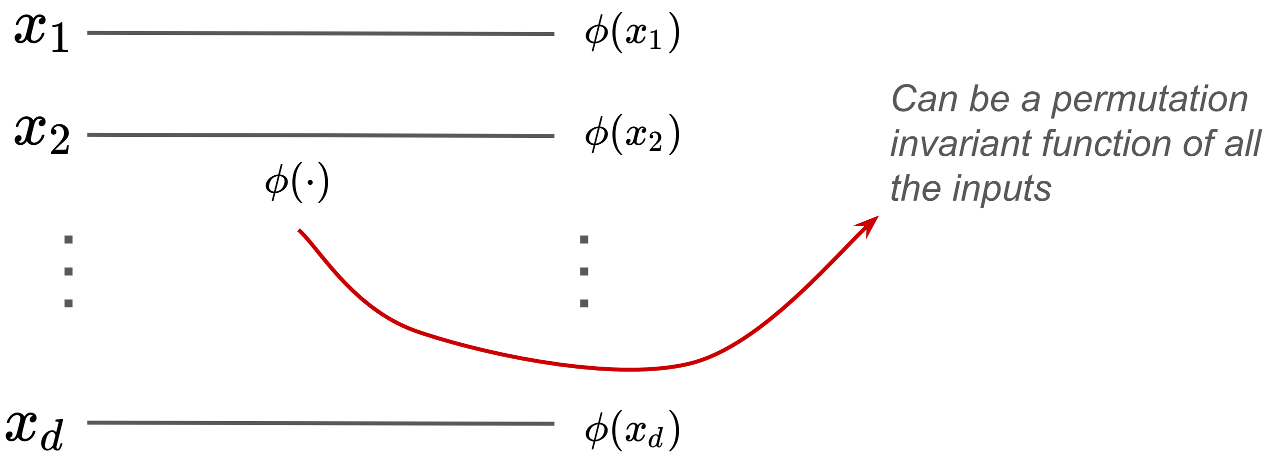
Equivariance over Sets

$\Omega = [d]$ $\qquad \mathcal{X}(\Omega) = \mathbb{R}^d$ $\qquad G = \{P \mid P = d \times d \text{ permutation matrix}\}$



Equivariance over Sets

$\Omega = [d]$ $\mathcal{X}(\Omega) = \mathbb{R}^d$ $G = \{P \mid P = d \times d \text{ permutation matrix}\}$



Recall: Attention and Transformer

is a permutation equivariant function over sets

$$x'_j = \sum_i \rho(\phi(x_j)^T \psi(x_i)) \cdot \alpha(x_i)$$

Invariance over Sets

$\Omega = [d]$ $\quad\quad$ $\mathcal{X}(\Omega) = \mathbb{R}^d$ $\quad\quad$ $G = \{P \mid P = d \times d \text{ permutation matrix}\}$

$$f(P \cdot x) = f(x)$$

Invariance over Sets

$\Omega = [d]$ $\mathcal{X}(\Omega) = \mathbb{X}^d$ $G = \{P \mid P = d \times d \text{ permutation matrix}\}$

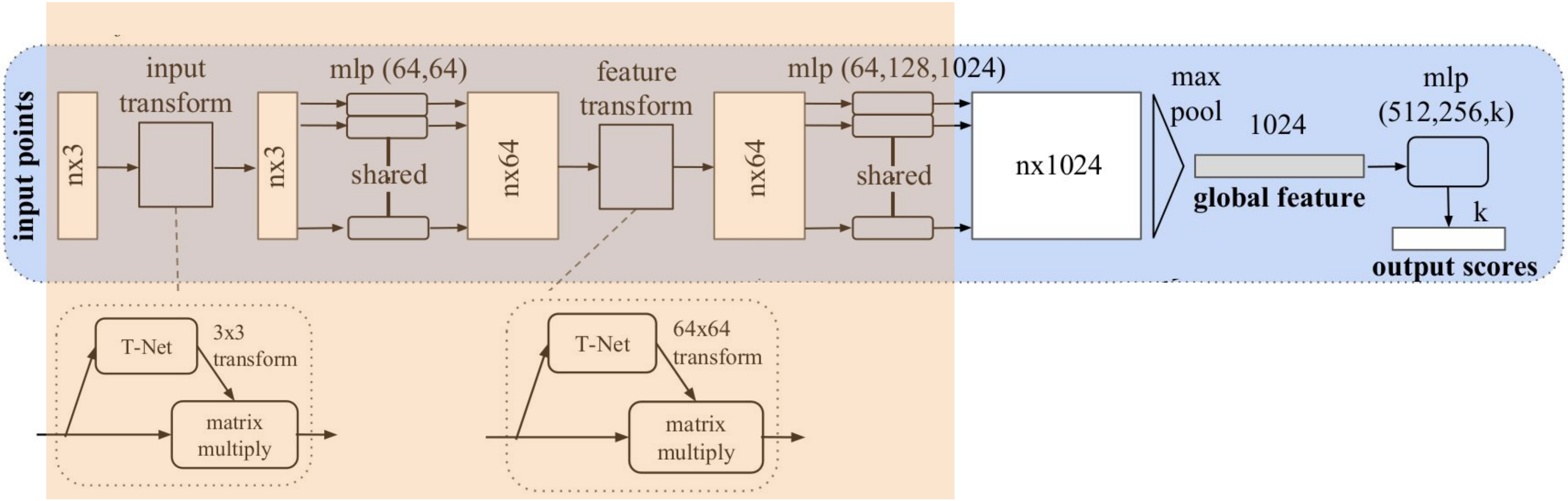
countable set

Theorem [Zaheer'17]:

$f : \mathcal{X}(\Omega) \to \mathbb{R}$ is G-invariant $\Longleftrightarrow$ $f(x) = \psi\left(\bigoplus_{i=1}^{d} \phi(x_i)\right)$ $\exists\, \phi, \psi$

Zaheer et al. "Deep Sets" NeurIPS 2017

Recall: PointNet Architecture



$$f(\{x_1, x_2, \dots x_n\}) = \max\{h(x_i), h(x_2), \dots h(x_n)\}$$

Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation” CVPR 2017

Simple Example

Permutation equivariant single layer perceptron

Question

Why not use the blueprint with permutation invariant/equivariant single layer perceptron?

Using G-invariant and G-equivariant single layer perceptrons

Zaheer's approximation

$$f(x) = \psi\left(\bigoplus_{i=1}^{d} \phi(x_i)\right)$$

Question

Why not use the blueprint with permutation invariant/equivariant single layer perceptron?

Using G-invariant and G-equivariant single layer perceptrons

Are invariant/equivariant

Zaheer's approximation

$$f(x) = \psi\left(\bigoplus_{i=1}^{d} \phi(x_i)\right)$$

Can approximate any G-invariant function

Graphs

Domain, Signals, Symmetry, and Function Spaces

$$\Omega = G = (V = [d], E)$$

$$\mathcal{X}(\Omega) = \left(\mathbb{R}^d, \mathcal{A}_G\right)$$

Space of all adjacency matrices on graph G

$$G = \{P \mid P = d \times d \text{ permutation matrix}\}$$

Domain, Signals, Symmetry, and Function Spaces

$$\Omega = G = (V = [d], E)$$

$$\mathcal{X}(\Omega) = (\mathbb{R}^d, \mathcal{A}_G)$$

Space of all adjacency matrices on graph G

$$G = \{P \mid P = d \times d \text{ permutation matrix}\}$$

$$\mathcal{F}^{\text{inv}} = \{f : \mathcal{X}(\Omega) \to \mathbb{R} \mid f(Px, PAP^T) = f(x, A)\}$$

$$\mathcal{F}^{\text{eqv}} = \{f : \mathcal{X}(\Omega) \to \mathbb{R} \mid f(Px, PAP^T) = P \cdot f(x, A)\}$$

Constructing Permutation Invariant Function

Easy! Any guesses?

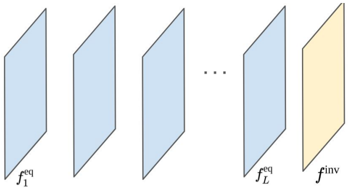
Constructing Permutation Equivariant Functions

Local function that operates over node neighborhoods

Local function must be invariant to order of neighbors

$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} - & \boxed{\phi}(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & - \\ - & \phi(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) & - \\ & \vdots & \\ - & \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) & - \end{bmatrix}$$

Applying the Blueprint



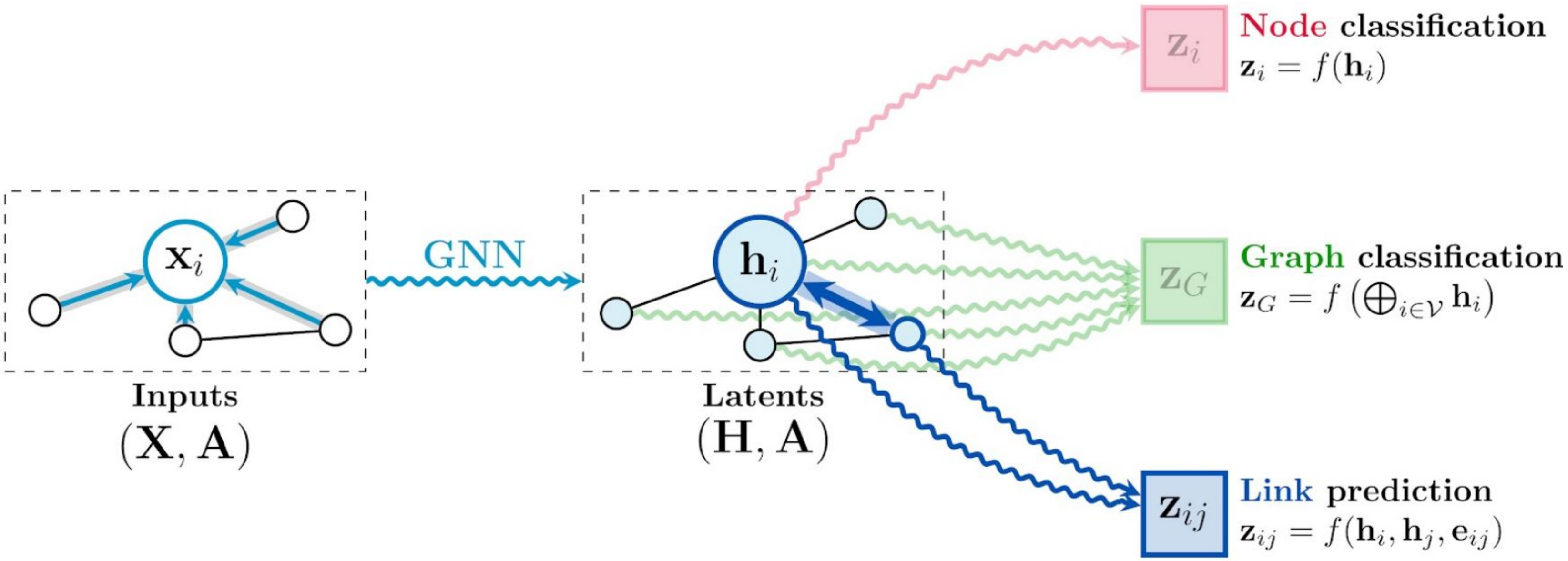
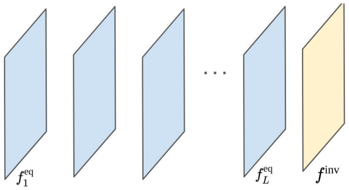


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Applying the Blueprint



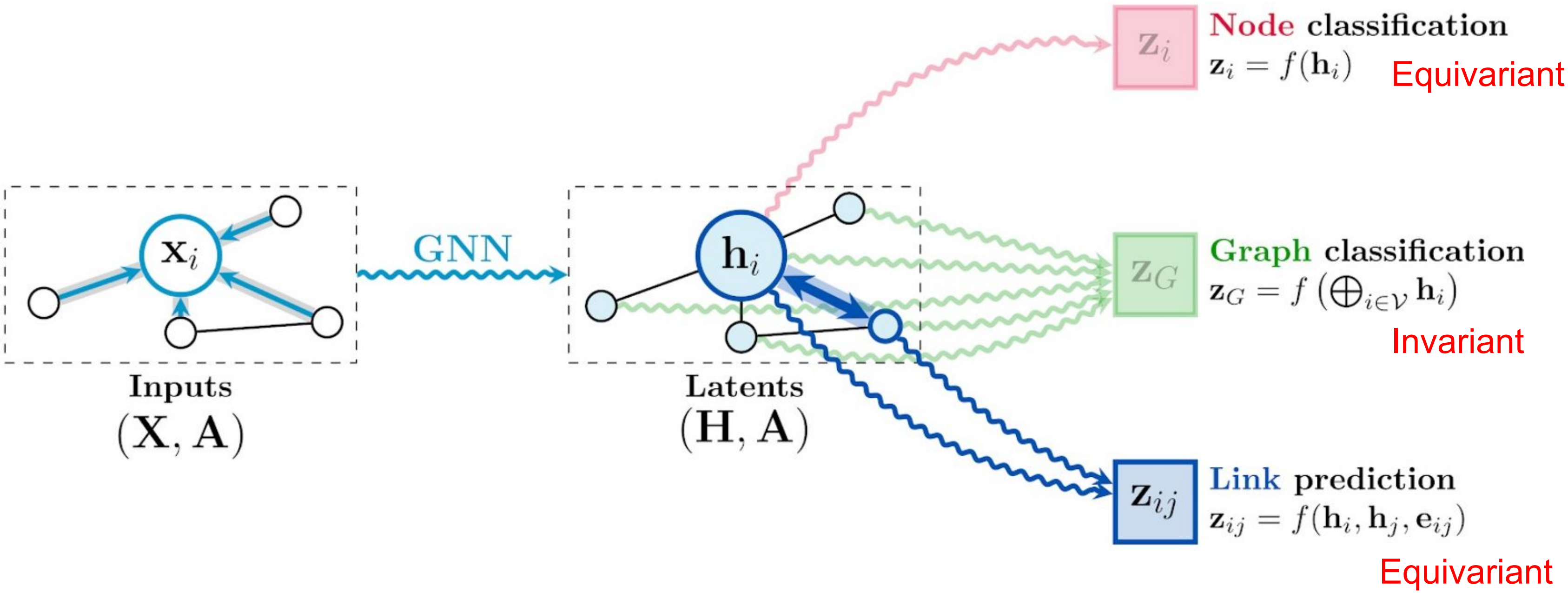


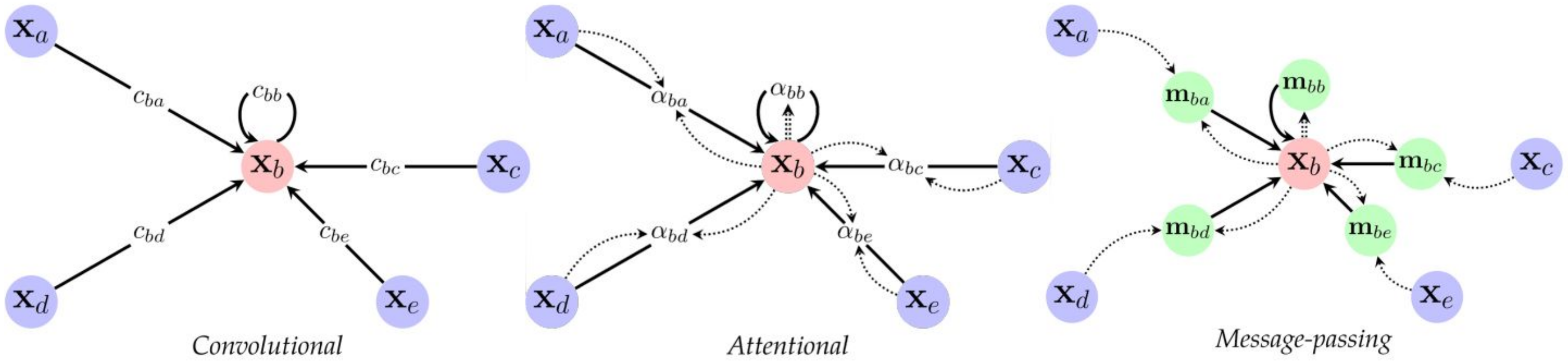
image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Constructing Permutation Equivariant Functions

How to construct these local functions?

$$\mathbf{F}(\mathbf{X},\mathbf{A}) = \begin{bmatrix} - & \boxed{\phi}(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & - \\ - & \phi(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) & - \\ & \vdots & \\ - & \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) & - \end{bmatrix}$$

Popular Graph Neural Networks



$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

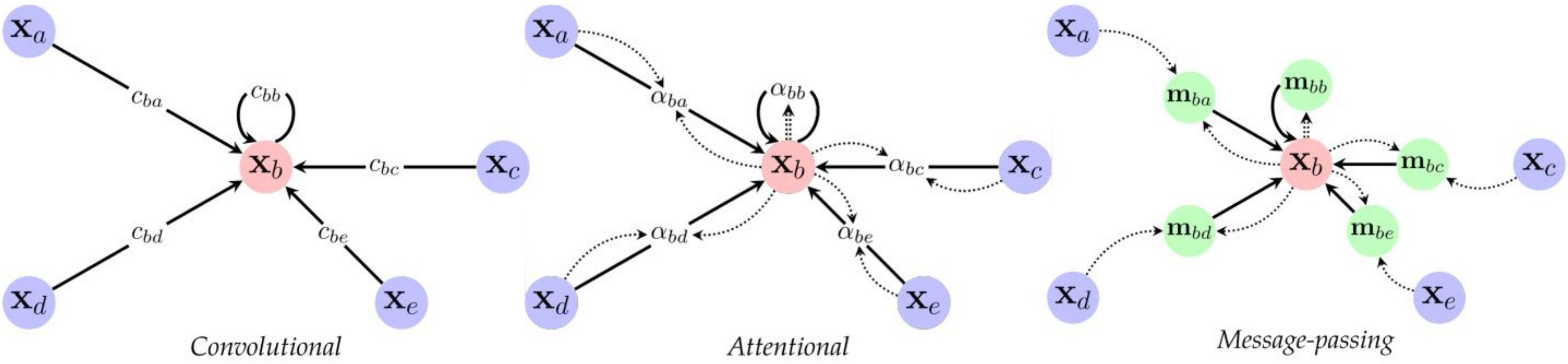
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Popular Graph Neural Networks

PyTorch Geometric

https://www.pytorch-geometric.readthedocs.io/



$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Constructing Equivariant and Invariant GNN

Node, edge, and graph features

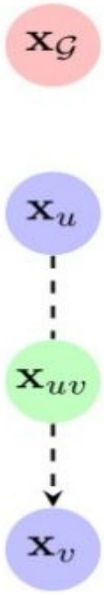
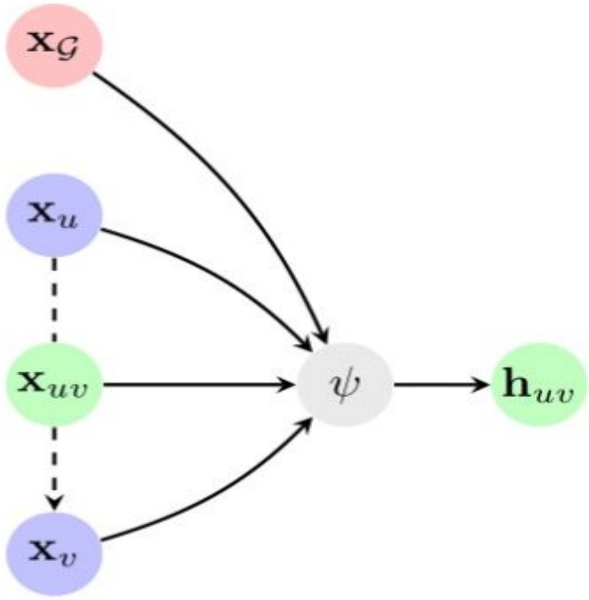


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Constructing Equivariant and Invariant GNN

Node, edge, and graph features

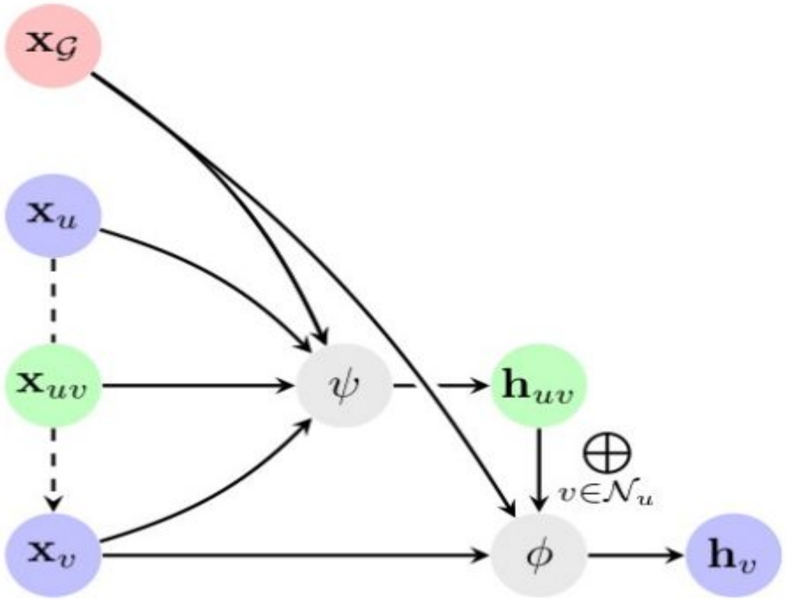


$$\mathbf{h}_{uv} = \psi(\mathbf{x}_u, \mathbf{x}_v, \mathbf{x}_{uv}, \mathbf{x}_\mathcal{G})$$

image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Constructing Equivariant and Invariant GNN

Node, edge, and graph features

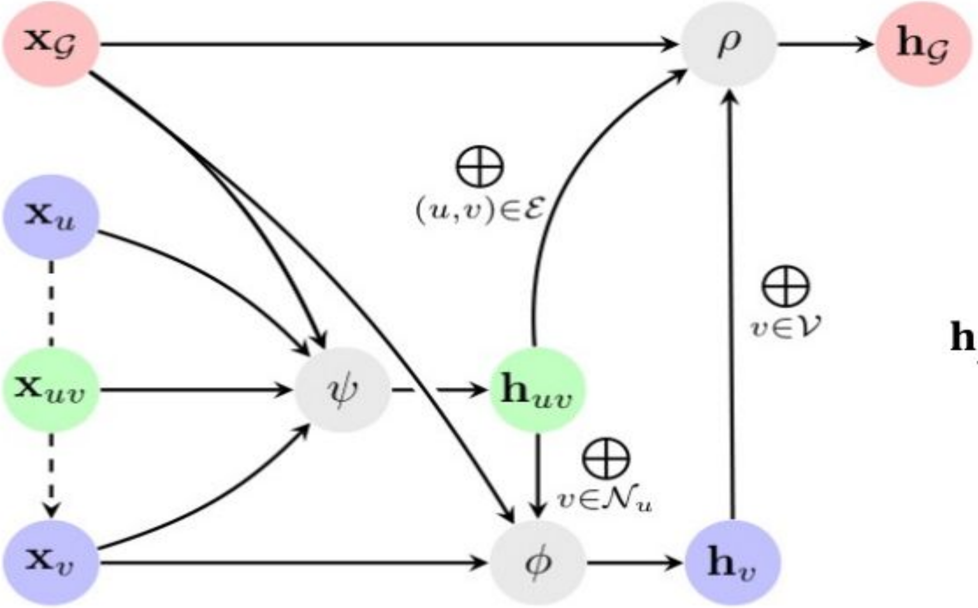


$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{u \in \mathcal{N}_v} \mathbf{h}_{vu}, \mathbf{x}_{\mathcal{G}}\right)$$

image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Constructing Equivariant and Invariant GNN

Node, edge, and graph features

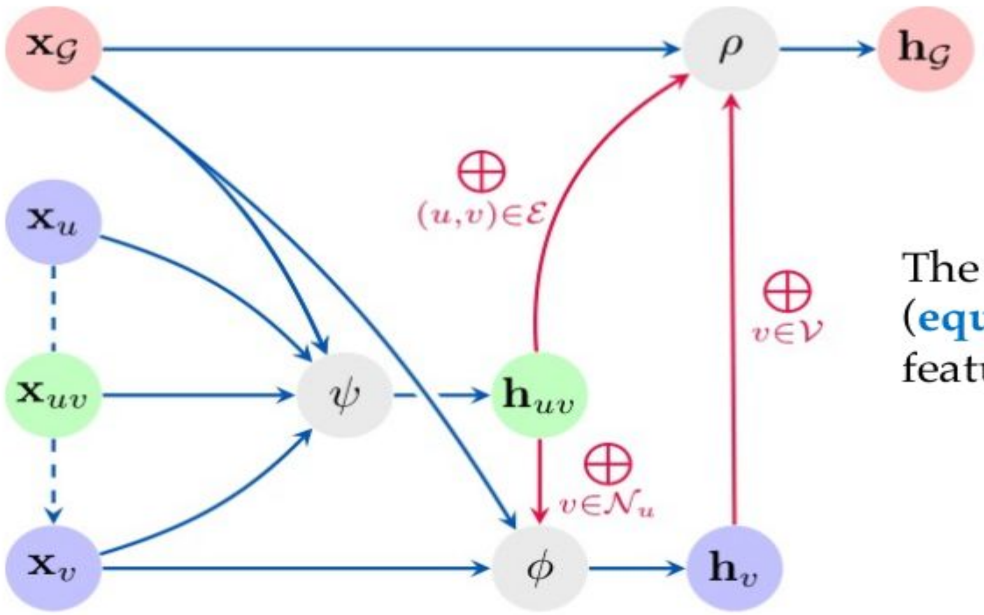


$$\mathbf{h}_{\mathcal{G}} = \rho\left(\bigoplus_{u\in\mathcal{V}} \mathbf{h}_u, \bigoplus_{(u,v)\in\mathcal{E}} \mathbf{h}_{uv}, \mathbf{x}_{\mathcal{G}}\right)$$

image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Constructing Equivariant and Invariant GNN

Node, edge, and graph features



The *geometric deep learning blueprint* (**equivariant** and **invariant** layers) features extensively in GNs

image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

image source: Krishna et al. "Visual Genome" 2016

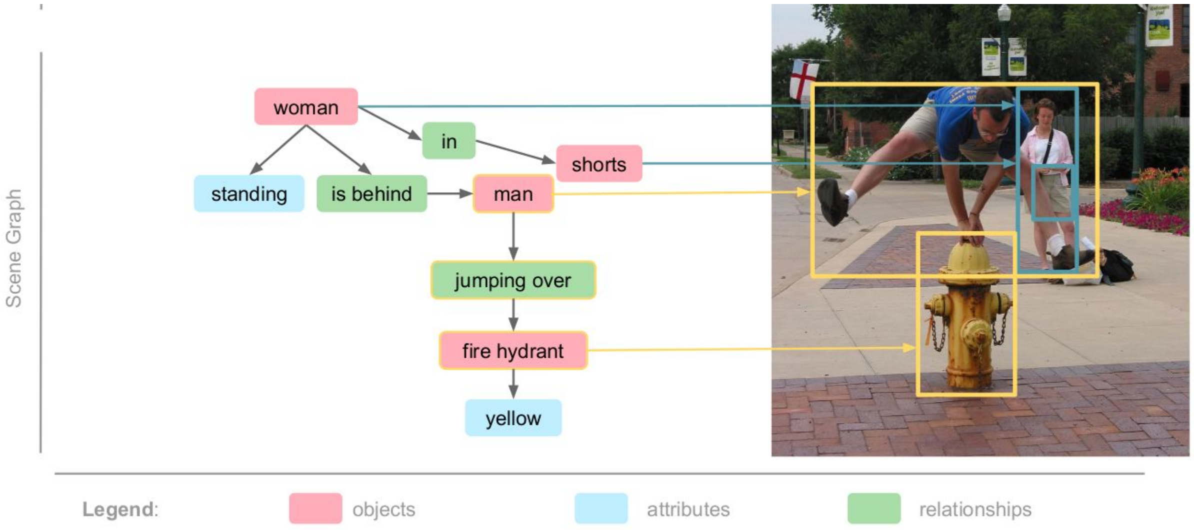
Scene Graphs

$$SG = (O, R, E)$$

objects

relations

edges



$$O = \{o_i = (c_i, a_i) \mid c_i = \text{class}, a_i = \text{attribute}\}$$

$$R = \{r_i \mid r_i = \text{relation}\}$$

$$E \subset O \times R \times O$$

$$e = (s, p, o)$$

subject-predicate-object

image source: Krishna et al. “Visual Genome” 2016

Scene Graphs

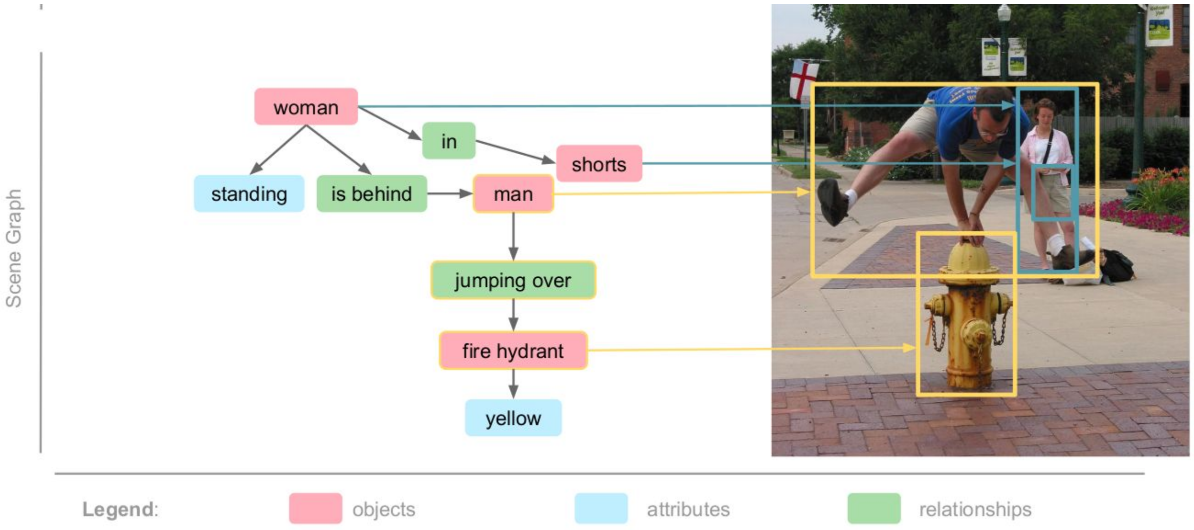
$$SG = (O, R, E)$$

objects

relations

edges

$$G = (O, E)$$



$$O = \{o_i = (c_i, a_i) \mid c_i = \text{class}, a_i = \text{attribute}\}$$

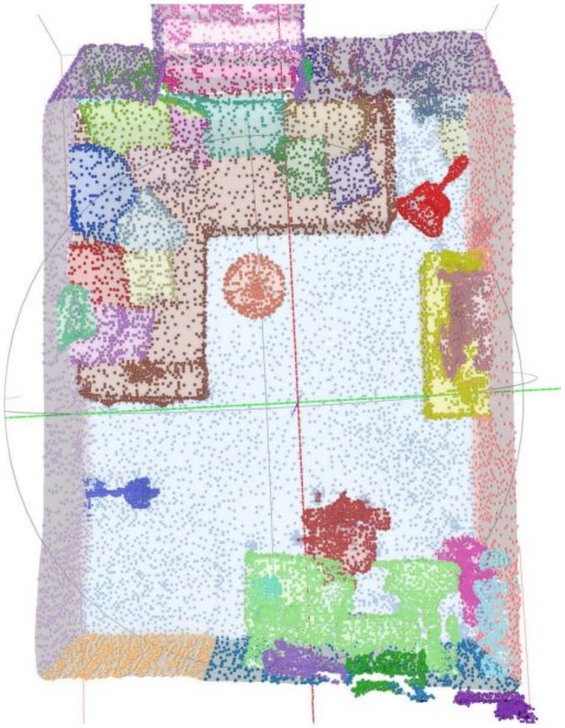
$$R = \{r_i \mid r_i = \text{relation}\}$$

$$E \subset O \times R \times O$$

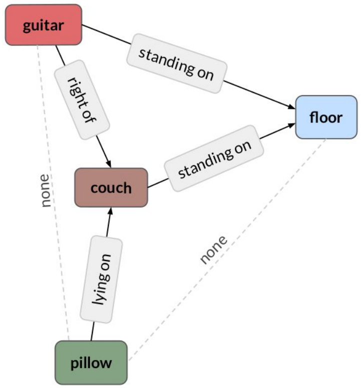
$$e = (s, p, o)$$

subject-predicate-object

Scene Graph Generation



Given a segmented 3D scene (voxel, point cloud, mesh), construct a scene graph



Problems

Node class and attribute labeling

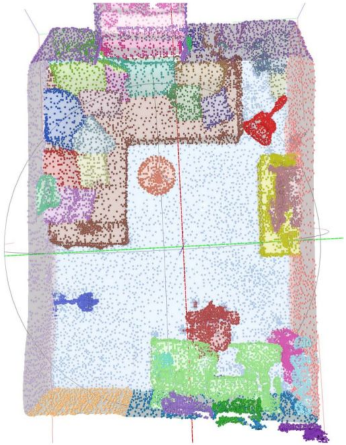
Relationship prediction and labeling

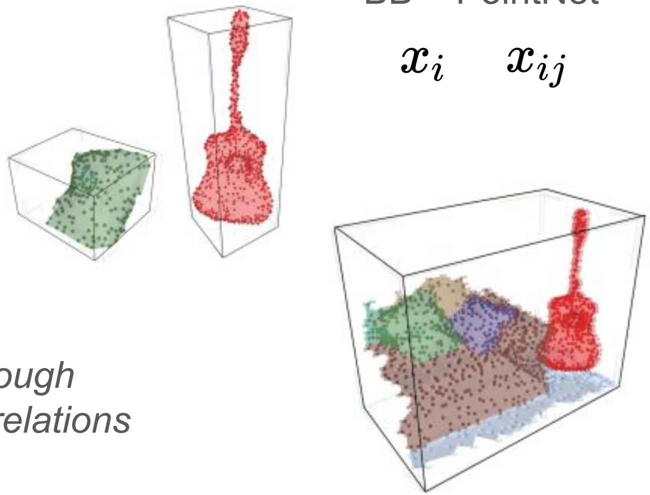
image source: Wald et al. “Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions” CVPR 2020

Two Approaches

- Conditional random field based methods
- Graph neural network based methods

But first ...





BB + PointNet

x_i x_{ij}

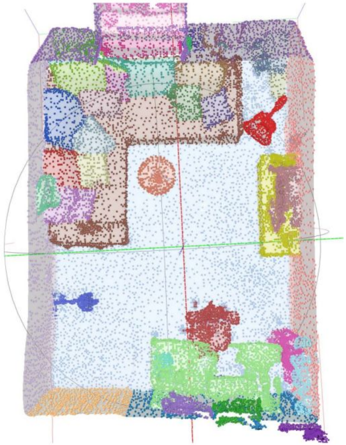
Need to extract expressive enough input features for objects and relations

image source: Wald et al. “Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions” CVPR 2020

Two Approaches

- Conditional random field based methods
- Graph neural network based methods

But first ...



Need to extract expressive enough input features for objects and relations

BB + PointNet

x_i x_{ij}

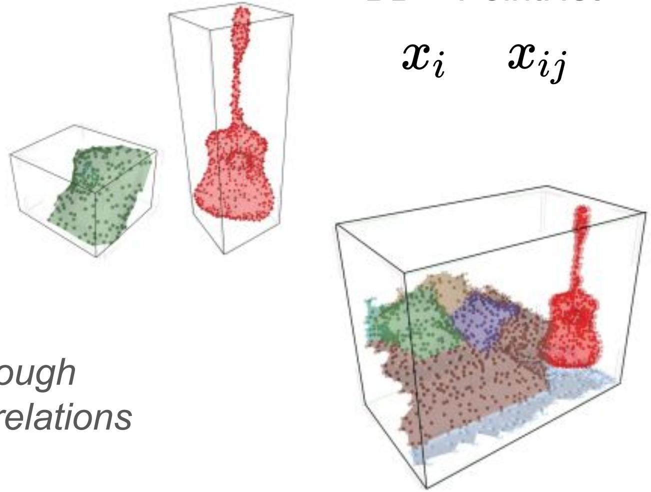
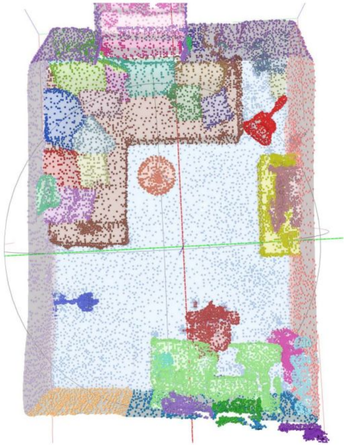


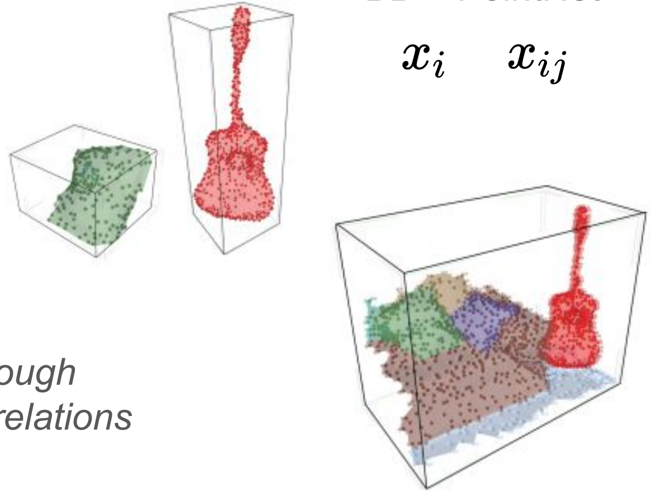
image source: Wald et al. “Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions” CVPR 2020

CRF-based Approaches

Function parameterized by W

$$P(r_e = r \mid x_{s_e}, x_{r_e}, x_{o_e}) = \frac{1}{Z}\exp\{\Phi(r \mid x_{s_e}, x_{r_e}, x_{o_e}, W)\}$$





BB + PointNet

x_i x_{ij}

Need to extract expressive enough input features for objects and relations

image source: Wald et al. “Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions” CVPR 2020

CRF-based Approaches

Function parameterized by W

$$P(r_e = r \mid x_{s_e}, x_{r_e}, x_{o_e}) = \frac{1}{Z}\exp\{\Phi(r \mid x_{s_e}, x_{r_e}, x_{o_e}, W)\}$$

Limitation

Does not take into account the spatial correlations between different objects and their relationship in the scene

GNN-based Approaches

1. x_s, x_r, x_o $\xrightarrow{\text{MLP}}$ h_s, x'_r, h_o

2. $$h'_i = \psi\left(\bigoplus_{j\,:\,(i-r-j)\in E} h_j, \bigoplus_{j\,:\,(j-r-i)\in E} h_j\right)$$

Propagate message across the graph and help learn spatial correlations

3. $$x'_i = x_i + \mathrm{MLP}(h'_i)$$

Wald et al. "Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions" CVPR 2020

Problems

Limited expressivity of graph neural networks

Limited Expressivity of Graph Neural Networks

Cannot distinguish between two different graphs.

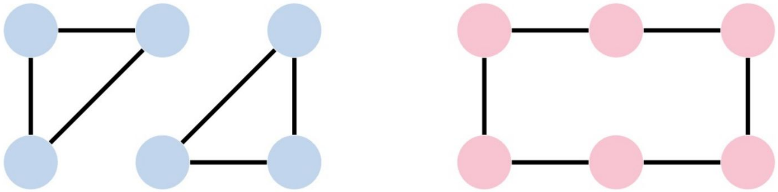


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Limited Expressivity of Graph Neural Networks

Cannot distinguish between two different graphs.

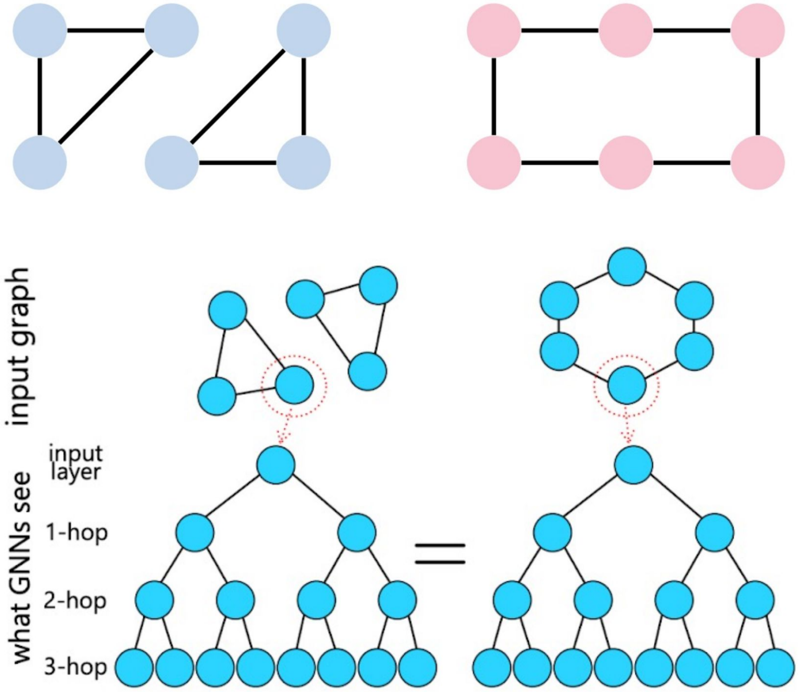


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Limited Expressivity of Graph Neural Networks

Graph Isomorphism Problem

Give two finite graphs G and H, determine if they are isomorphic

- *Hard problem to solve.*
- *Not known if polynomial time or NP-complete.*
- *Complexity exponential in graph treewidth.*

Weisfeiler-Lehman Test for Graph Isomorphism

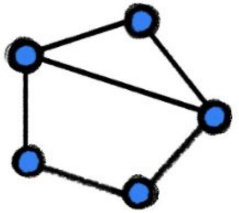


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Weisfeiler-Lehman Test for Graph Isomorphism

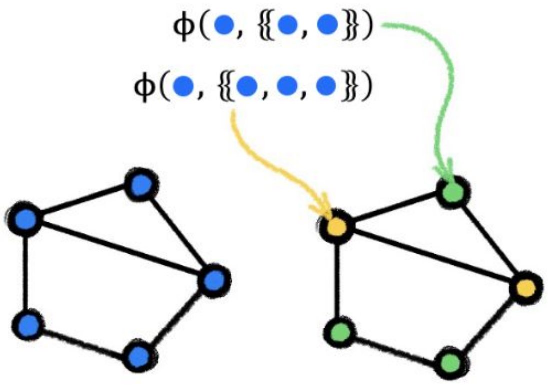


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Weisfeiler-Lehman Test for Graph Isomorphism

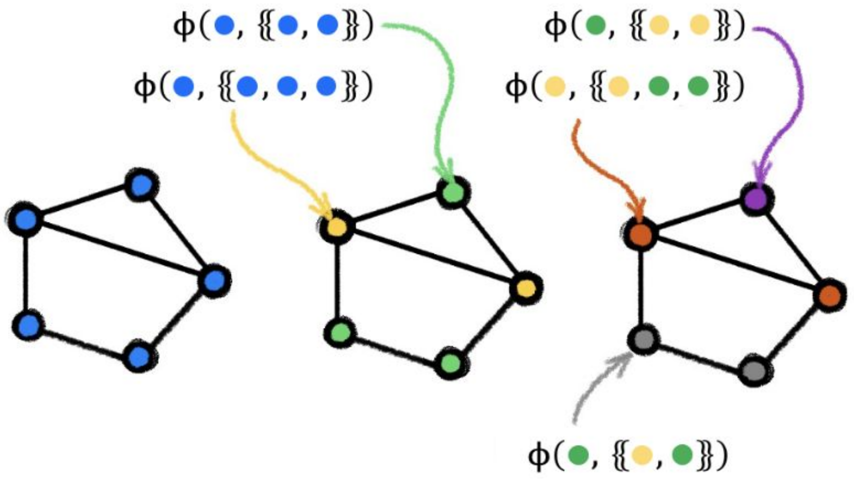


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Weisfeiler-Lehman Test for Graph Isomorphism

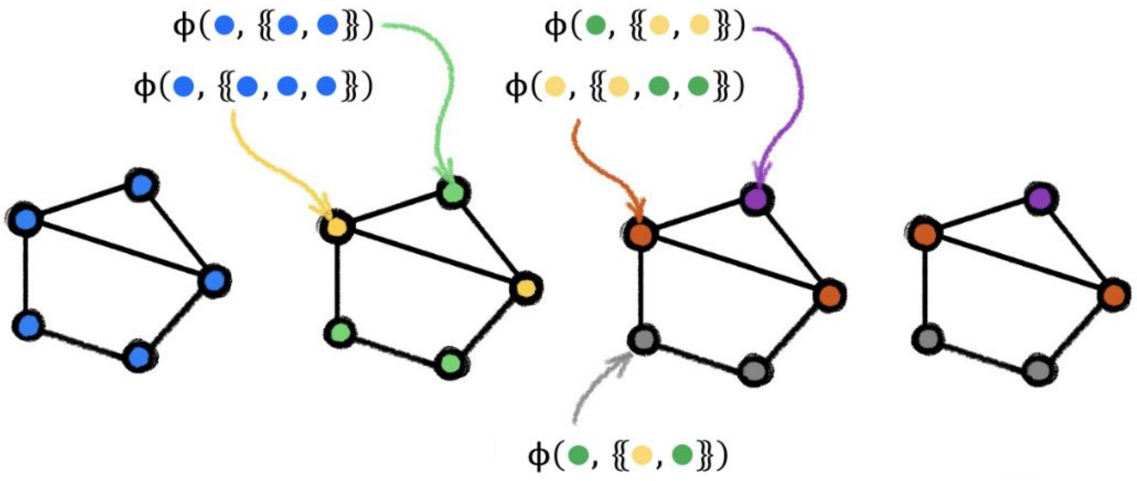


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Weisfeiler-Lehman Test for Graph Isomorphism

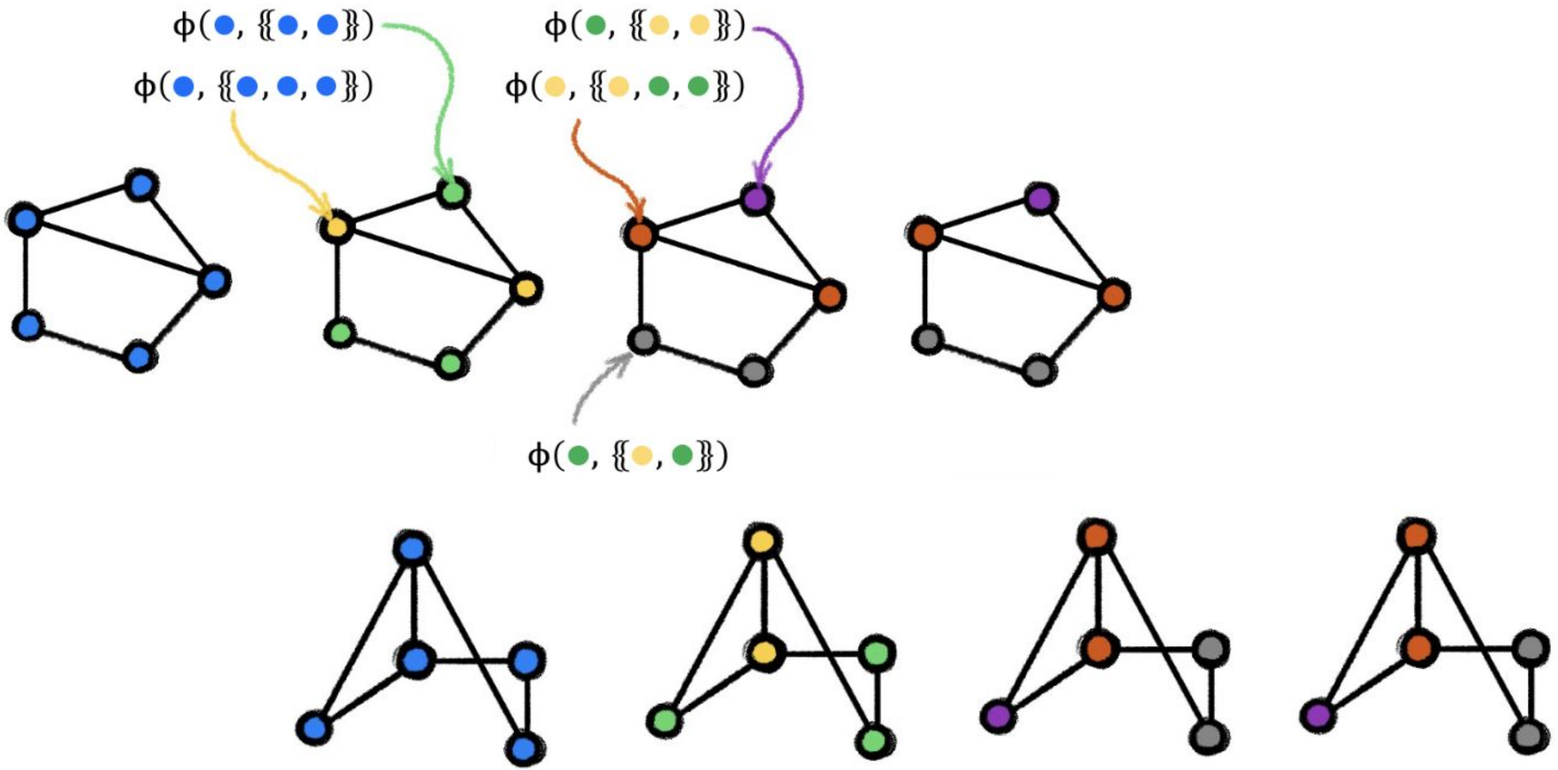
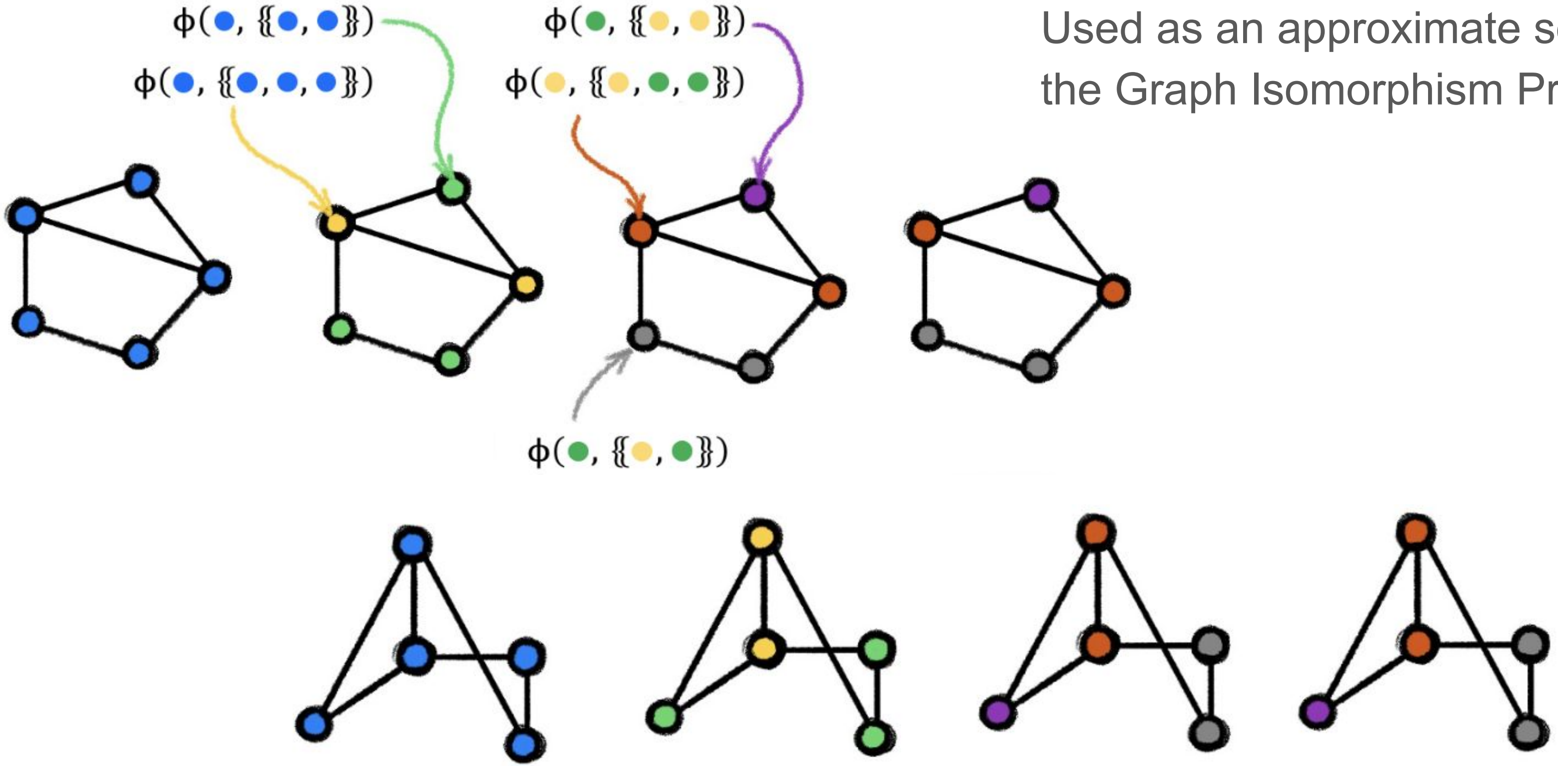


image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Weisfeiler-Lehman Test for Graph Isomorphism



Used as an approximate solution to the Graph Isomorphism Problem

image source: Bronstein et al. “Geometric Deep Learning” Lectures for AMMI, 2021.

Limited Expressivity of Graph Neural Networks

- Expressivity of graph neural networks is less than the Weisfeiler-Lehman test.
- For discrete feature space, graph isomorphism network does as well as the Weisfeiler-Lehman test.

$$x'_v = \mathrm{MLP}\left((1+\epsilon)x_v + \sum_{u \in N_v} x_u\right)$$

Xu et al. "How Powerful are Graph Neural Networks?" ICLR 2019

- Does not hold for continuous feature space!

Why?

Improving Expressivity of Graph Neural Networks

Sets

Using G-invariant and G-equivariant single layer perceptrons

Zaheer's approximation

$$f(x) = \psi\left(\bigoplus_{i=1}^{d} \phi(x_i)\right)$$

Graphs

Standard Graph Neural Networks

Improving Expressivity of Graph Neural Networks

Sets

Using G-invariant and G-equivariant single layer perceptrons

Are invariant/equivariant

Zaheer's approximation

$$f(x) = \psi\left(\bigoplus_{i=1}^{d} \phi(x_i)\right)$$

Can approximate any G-invariant function

Graphs

Standard Graph Neural Networks

Are invariant/equivariant

Improving Expressivity of Graph Neural Networks

Sets

Using G-invariant and G-equivariant single layer perceptrons

Are invariant/equivariant

Zaheer's approximation

$$f(x) = \psi\left(\bigoplus_{i=1}^{d} \phi(x_i)\right)$$

Can approximate any
G-invariant function

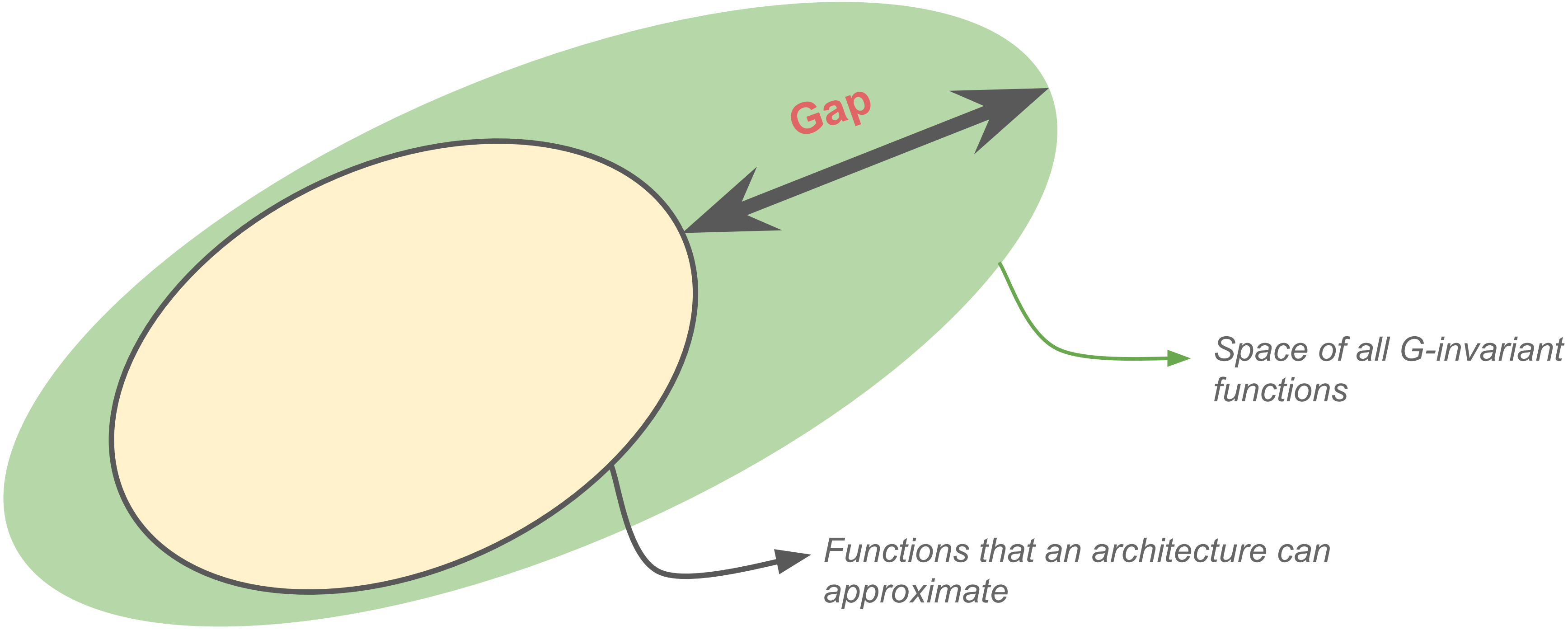
Graphs

Standard Graph Neural Networks

Are invariant/equivariant

Architecture that can approximate
any G-invariant/equivariant function

Improving Expressivity of Graph Neural Networks

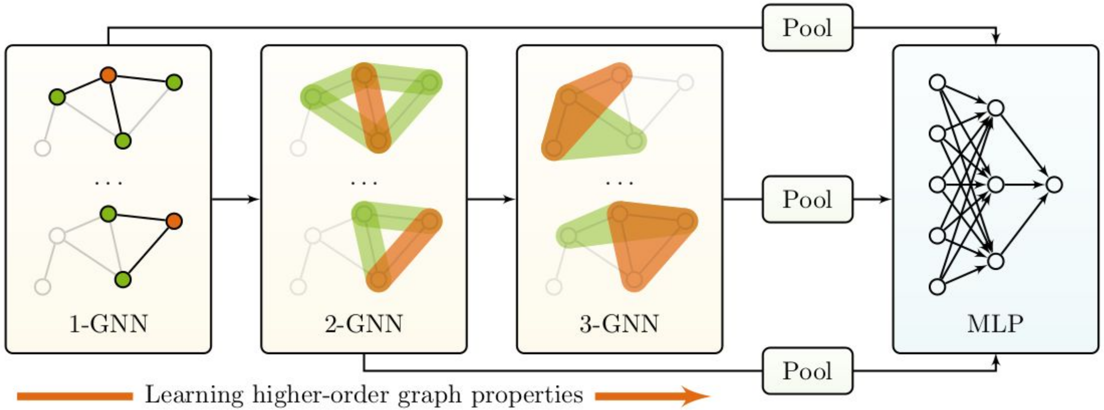


k-order Graph Neural Network

- Form a k-order Graph

$$G = (V, E) \longrightarrow G^k = (V^k, \tilde{E})$$

- Graph Neural Network on k-order graphs



Morris et al. "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks" 2019

k-order Graph Neural Network

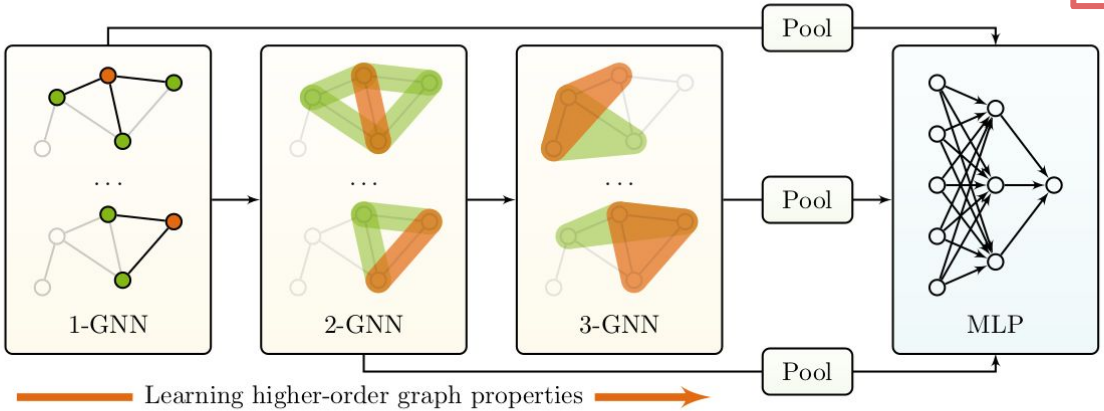
- Form a k-order Graph

$$G = (V, E) \longrightarrow G^k = (V^k, \tilde{E})$$

Can approximate any G-invariant function as

$$k \rightarrow \infty$$

- Graph Neural Network on k-order graphs



Morris et al. “Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks” 2019

Improving Expressivity of Graph Neural Networks

Using G-invariant and G-equivariant single layer perceptrons	Standard Graph Neural Networks
Are invariant/equivariant	Are invariant/equivariant
Zaheer's approximation	↓ **Ongoing Research**
Can approximate any G-invariant function	Architecture that can approximate **any** G-invariant/equivariant function

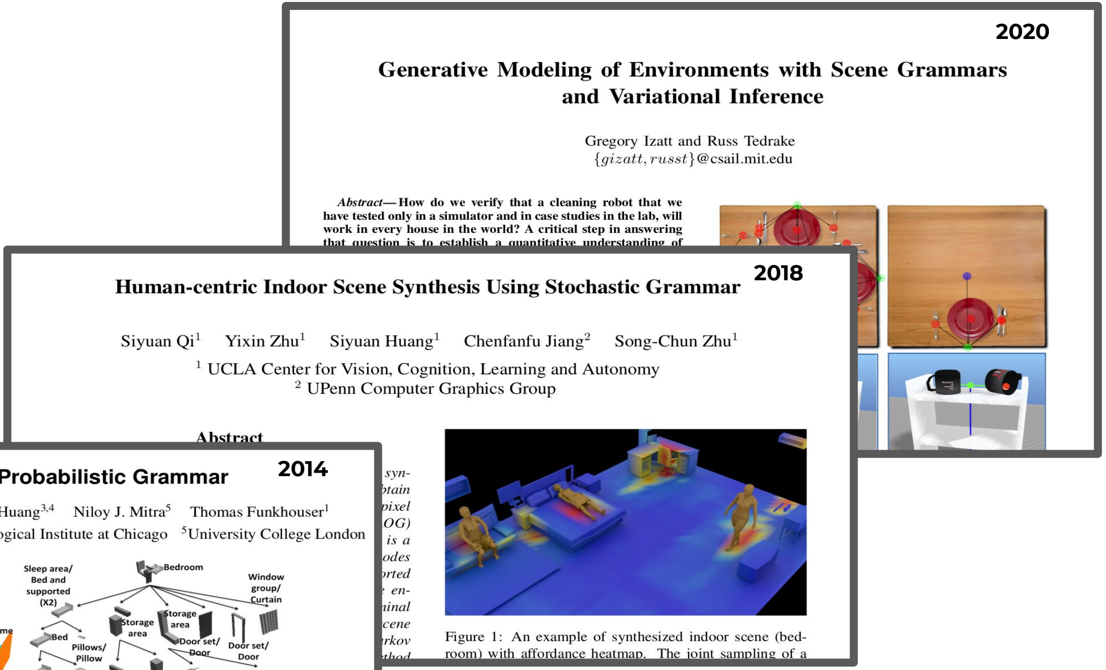
Models for Scene Graphs

Probabilistic graphical models have been used to describe scene graphs

$$p(\mathbf{X}|\mathcal{G}) = \frac{1}{Z}\prod_{C}\psi_C(\mathbf{x}_C)$$

product of
clique potentials

Exact inference is NP-hard and
exponential in graph treewidth



2020

Generative Modeling of Environments with Scene Grammars and Variational Inference

Gregory Izatt and Russ Tedrake
{gizatt, russt}@csail.mit.edu

Abstract— How do we verify that a cleaning robot that we have tested only in a simulator and in case studies in the lab, will work in every house in the world? A critical step in answering that question is to establish a quantitative understanding of

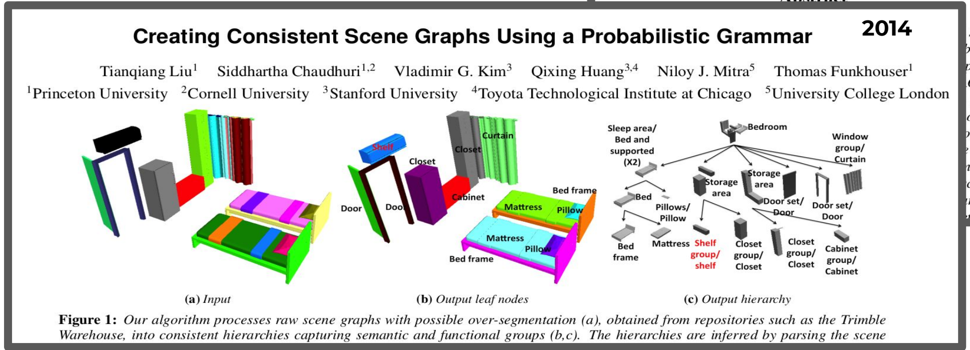
2018

Human-centric Indoor Scene Synthesis Using Stochastic Grammar

Siyuan Qi[1] Yixin Zhu[1] Siyuan Huang[1] Chenfanfu Jiang[2] Song-Chun Zhu[1]
[1] UCLA Center for Vision, Cognition, Learning and Autonomy
[2] UPenn Computer Graphics Group

Abstract

Figure 1: An example of synthesized indoor scene (bedroom) with affordance heatmap. The joint sampling of a



2014

Creating Consistent Scene Graphs Using a Probabilistic Grammar

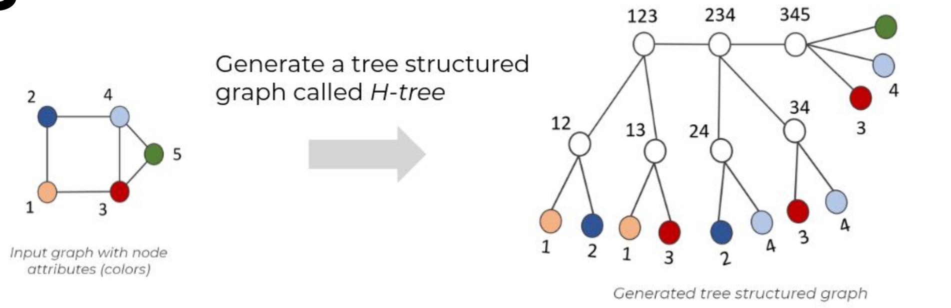
Tianqiang Liu[1] Siddhartha Chaudhuri[1,2] Vladimir G. Kim[3] Qixing Huang[3,4] Niloy J. Mitra[5] Thomas Funkhouser[1]
[1]Princeton University [2]Cornell University [3]Stanford University [4]Toyota Technological Institute at Chicago [5]University College London

(a) *Input* (b) *Output leaf nodes* (c) *Output hierarchy*

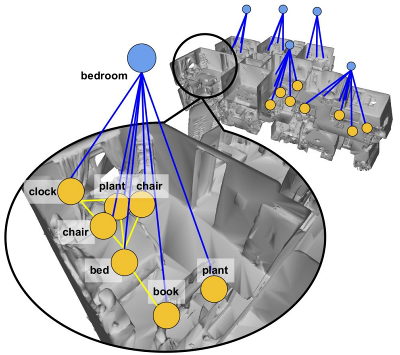
Figure 1: *Our algorithm processes raw scene graphs with possible over-segmentation (a), obtained from repositories such as the Trimble Warehouse, into consistent hierarchies capturing semantic and functional groups (b,c). The hierarchies are inferred by parsing the scene*

New: Neural Trees

- Neural Tree architecture
- Approximation Results
- Experiments

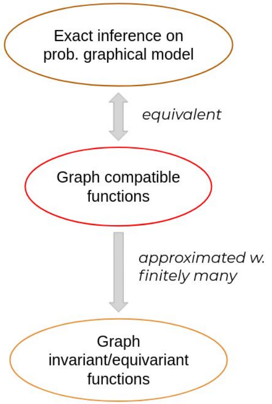


Neural Tree is *message passing on H-tree*



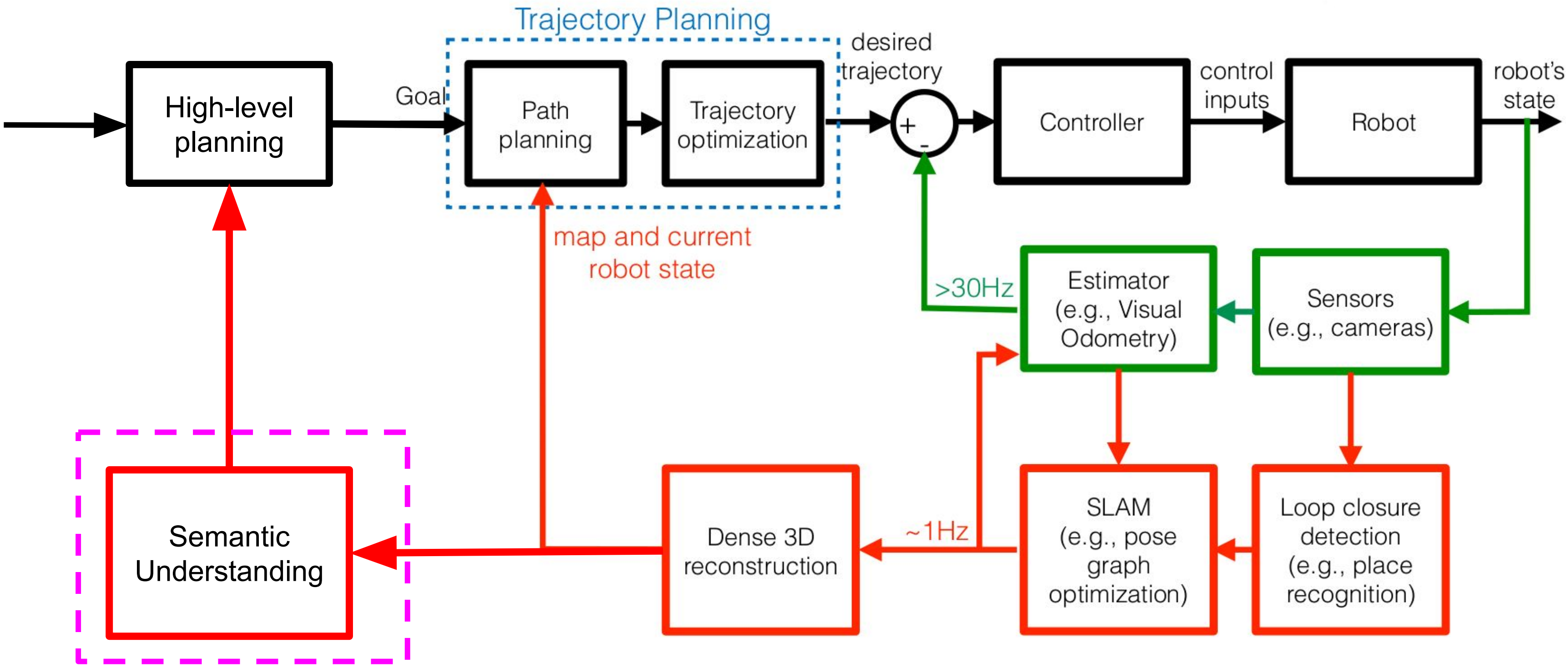
Any (smooth) graph compatible function can be approximated by a Neural Tree with number of weights/parameters

$$N = \mathcal{O}\left(\underbrace{n}_{\text{num. nodes}} \cdot \left(\underbrace{\mathrm{tw}(G)}_{\text{treewidth}} / \underbrace{\epsilon}_{\text{approx. distance}}\right)^{c \cdot \mathrm{tw}(G)}\right)$$



Rajat Talak, Siyi Hu, Lisa Peng, and Luca Carlone "**Neural Trees for Learning on Graphs**" NeurIPS 2021

Conclusion



Trajectory Planning
High-level planning
Goal
Path planning
Trajectory optimization
desired trajectory
+
-
Controller
control inputs
Robot
robot's state
map and current robot state
>30Hz
Estimator (e.g., Visual Odometry)
Sensors (e.g., cameras)
Semantic Understanding
Dense 3D reconstruction
~1Hz
SLAM (e.g., pose graph optimization)
Loop closure detection (e.g., place recognition)

Backup

Error Decomposition