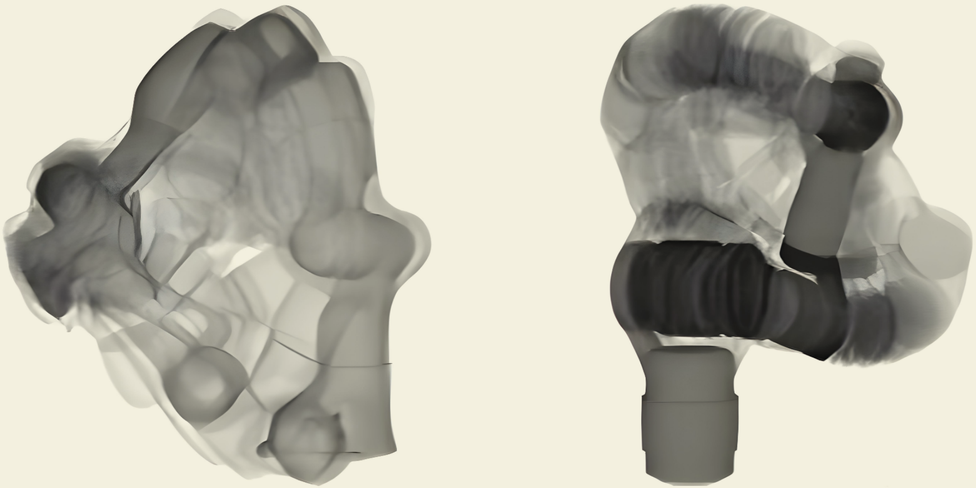
Generative Graphical Inverse Kinematics (GGIK)

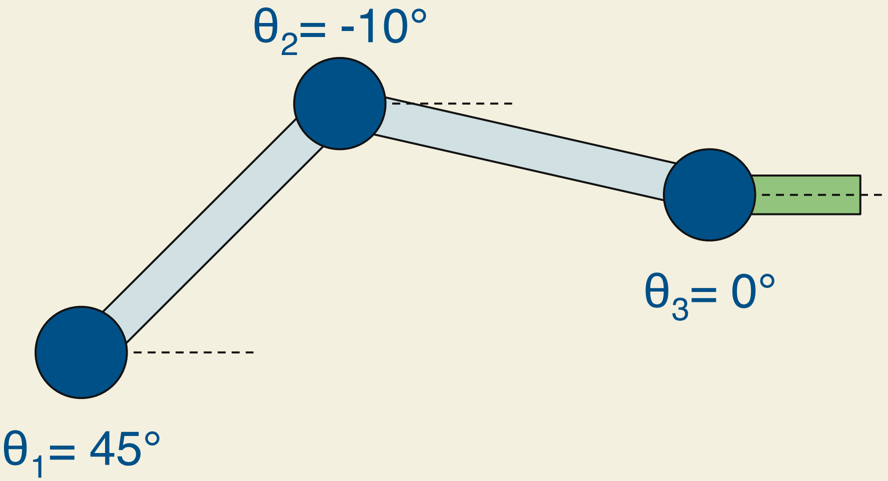
Calvin Stahoviak
Gabriel Urbaitis



Institution: *University of New Mexico*
Course: *ST, Advance Machine Learning*
Professor: *Trilce Estrada*

1. **Problem Statement**
2. **GGIK Architecture**
3. **Improvements**
4. **Experimental Results**
5. **Drawing a Conclusion**

Forward kinematics (FK) is the process of calculating the position and orientation of a robot's end-effector from its joint angles.



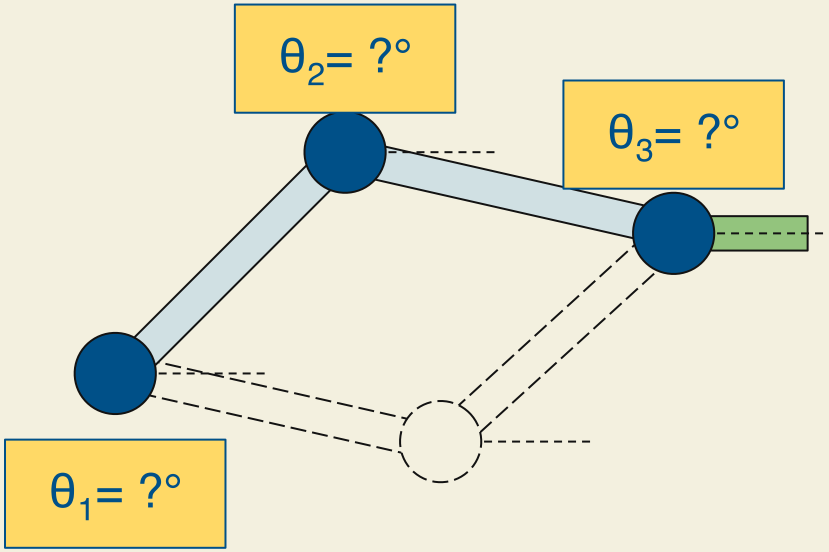
End-effector:

x = ?
y = ?
z = ?

The process of finding the joint angles needed to reach a desired position and orientation is called **<u>inverse kinematics</u>** (IK).

<u>End-effector:</u>

$x = 1.5$
$y = 0.4$
$z = 0.0$

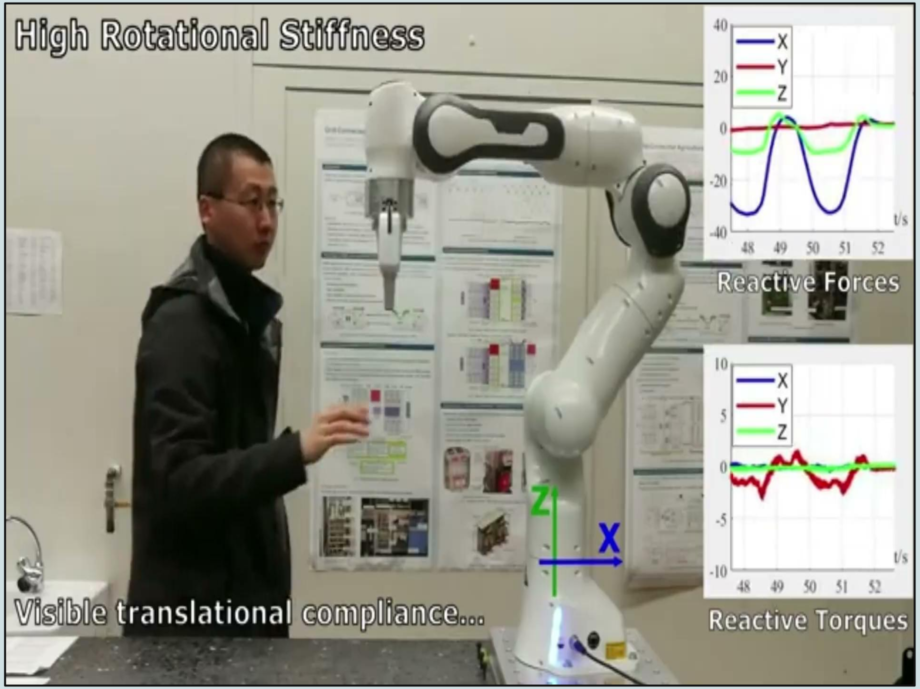


1. Problem Statement

Infinite Solutions

A robot arm with 7 or more degrees-of-freedom has infinite solutions

YouTube



Why is IK suitable for machine learning?

Non-Linear

- IK is highly non-linear
- **AI is well-suited to approximate complex mappings**

Fast Inference

- **Real time response is critical** for robots
- Forward pass is faster than iteration

Complex Objective Function

- **Numerical methods can fail** to minimize if improperly seeded

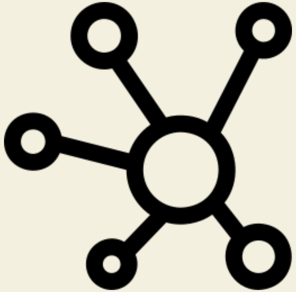
Addresses Redundancy

- Redundant robots have infinite IK solutions
- **Can learn a *distribution* of solutions**

1. Problem Statement
2. **GGIK Architecture**
3. Improvements
4. Experimental Results
5. Drawing a Conclusion

Robots To Graphs

How do we model our question as a graph?



Riemannian Optimization for Distance-Geometric Inverse Kinematics

Filip Marić, Matthew Giamou, Adam W. Hall, Soroush Khoubyarian, Ivan Petrović, Jonathan Kelly

IEEE Transactions on Robotics, 2022

Riemannian Optimization for Distance-Geometric Inverse Kinematics

Filip Marić, Matthew Giamou, Adam W. Hall, Soroush Khoubyarian, Ivan Petrović, and Jonathan Kelly

***Abstract*—Solving the inverse kinematics problem is a fundamental challenge in motion planning, control, and calibration for articulated robots. Kinematic models for these robots are typically parameterized by joint angles, generating a complicated mapping between the robot configuration and the end-effector pose. Alternatively, the kinematic model and task constraints can be represented using invariant distances between points attached to the robot. In this article, we formalize the equivalence of distance-based inverse kinematics and the distance geometry problem for a large class of articulated robots and task constraints. Unlike previous approaches, we use the connection between distance geometry and low-rank matrix completion to find inverse kinematics solutions by completing a partial Euclidean distance matrix through local optimization. Furthermore, we parameterize the space of Euclidean distance matrices with the Riemannian manifold of fixed-rank Gram matrices, allowing us to leverage a variety of mature Riemannian optimization methods. Finally, we show that bound smoothing can be used to generate informed initializations without significant computational overhead, improving convergence. We demonstrate that our inverse kinematics solver achieves higher success rates than traditional techniques and substantially outperforms them on problems that involve many workspace constraints.**

***Index Terms*—Computational geometry, kinematics, motion and path planning, Riemannian optimization.**

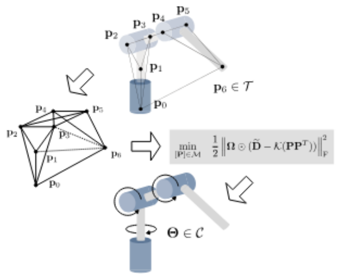


Fig. 1. Overview of the proposed algorithm. A goal end-effector position $\mathbf{p}_6$ is defined for a three-DOF robotic manipulator (top); the IK problem is to find the corresponding joint angles $\mathbf{\Theta}$. Our method uses the matrix $\tilde{\mathbf{D}}$ of distances between points $\mathbf{P}$ common to all feasible IK solutions to define an incomplete graph whose edges are weighted by known distances. Then, we apply EDM completion with the known distance selection matrix $\mathbf{\Omega}$ to recover the weights corresponding to the unknown edges, solving the IK problem.

I. Introduction

ARTICULATED robots consist of actuated revolute joints connected by rigid links. A significant portion of the difficulty associated with performing a specific task involves finding joint angles that achieve a desired end-effector pose. Identifying a set of joint angles or a *configuration* that reaches the desired goal pose(s) of one or more end-effectors is known as the *inverse kinematics* (IK) problem [1]. In general, this problem cannot be solved analytically and admits an infinite number of solutions for robots with redundant degrees of freedom (DOFs). Therefore, most approaches resort to numerical methods that solve constrained local optimization problems over joint angles. This leads to constraints on end-effector and link poses that

Manuscript received July 30, 2021; accepted October 7, 2021. Date of publication December 1, 2021; date of current version June 7, 2022. This article was recommended for publication by Associate Editor R. Murrieta-Cid and Editor E. Yoshida upon evaluation of the reviewers' comments. This work was supported in part by the European Regional Development Fund under Grant KK.01.1.1.01.0009 (DATACROSS) and in part by the Natural Sciences and Engineering Research Council of Canada. The work of Jonathan Kelly was supported by the Canada Research Chairs Program. *(Filip Marić and Matthew Giamou contributed equally to this work.) (Corresponding author: Filip Marić.)*

Filip Marić is with the Space and Terrestrial Autonomous Robotic Systems Laboratory, University of Toronto Institute for Aerospace Studies, Toronto, ON M3H 5T6, Canada, and also with the Laboratory for Autonomous Systems and Mobile Robotics, Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia (e-mail: filip.maric@robotics.utias.utoronto.ca).

Matthew Giamou, Soroush Khoubyarian, and Jonathan Kelly are with the Space and Terrestrial Autonomous Robotic Systems Laboratory, University of Toronto Institute for Aerospace Studies, Toronto, ON M3H 5T6, Canada (e-mail: matthew.giamou@mail.utoronto.ca; soroush.khoubyarian@mail.utoronto.ca; jkelly@utias.utoronto.ca).

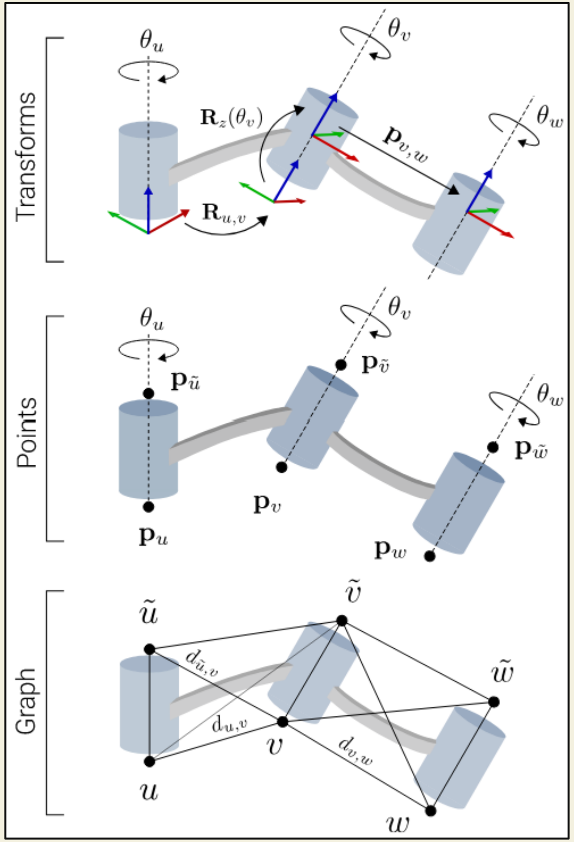
2. GGIK Architecture - Dataset Generation

Distance-Geometric Representations

1. *Place two points along the rotation axis of each joint.*

2. *For each consecutive pair of joints, connect all four points with edges*

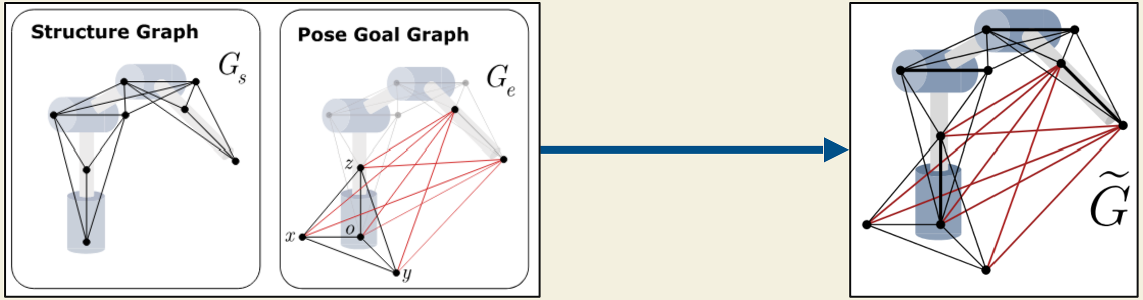
3. *Let edge weights be the Euclidean distance between points*



Input & Output Graphs

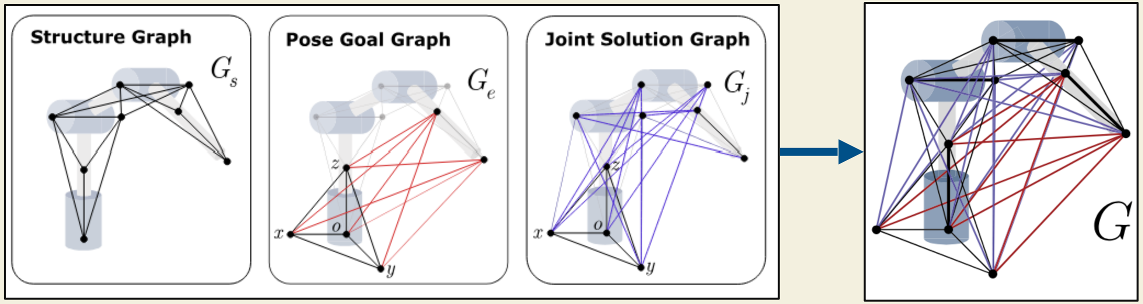
A partial graph $\tilde{G}$ represents the question/input:

$$\tilde{G} = G_s \cup G_e$$



A complete graph G represents the answer/output:

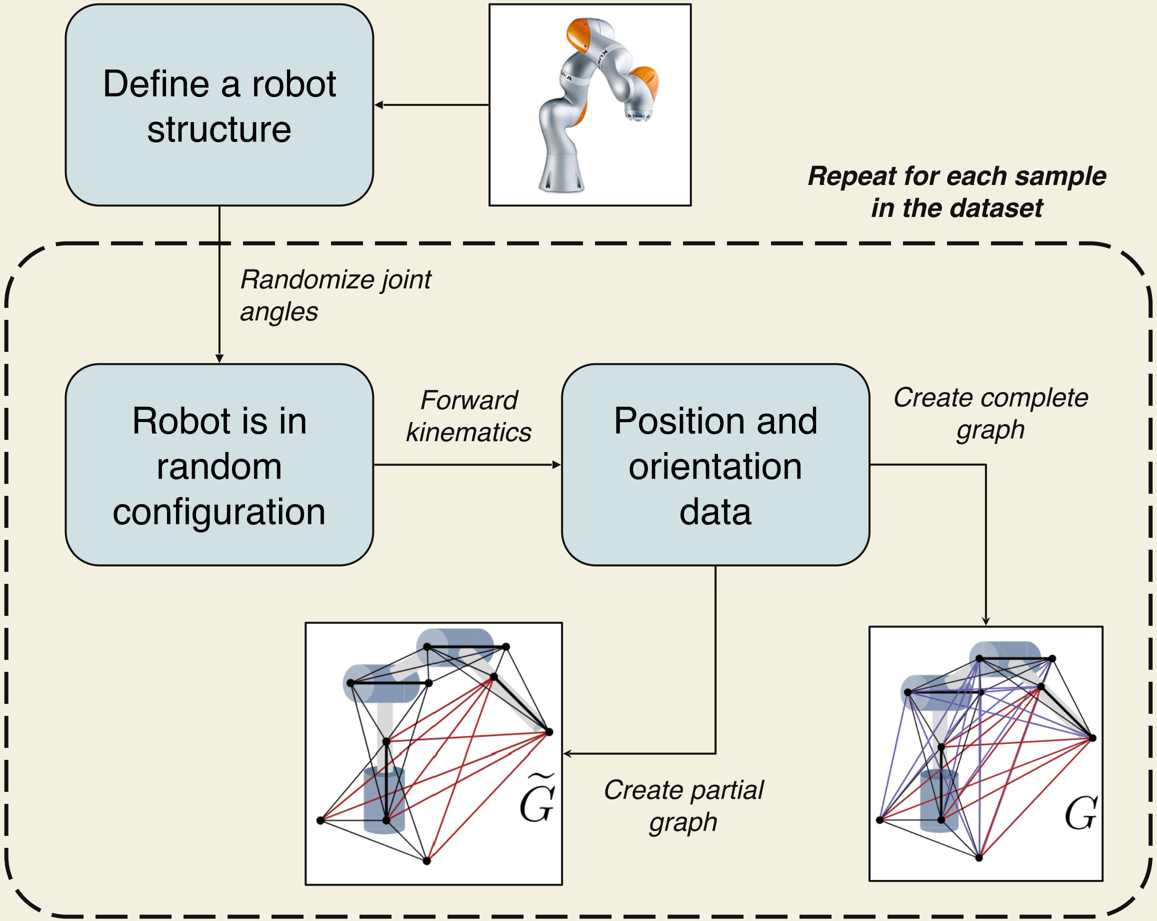
$$G = G_s \cup G_e \cup G_j$$



Dataset Generation

*Use **forward kinematics** to generate a dataset:*

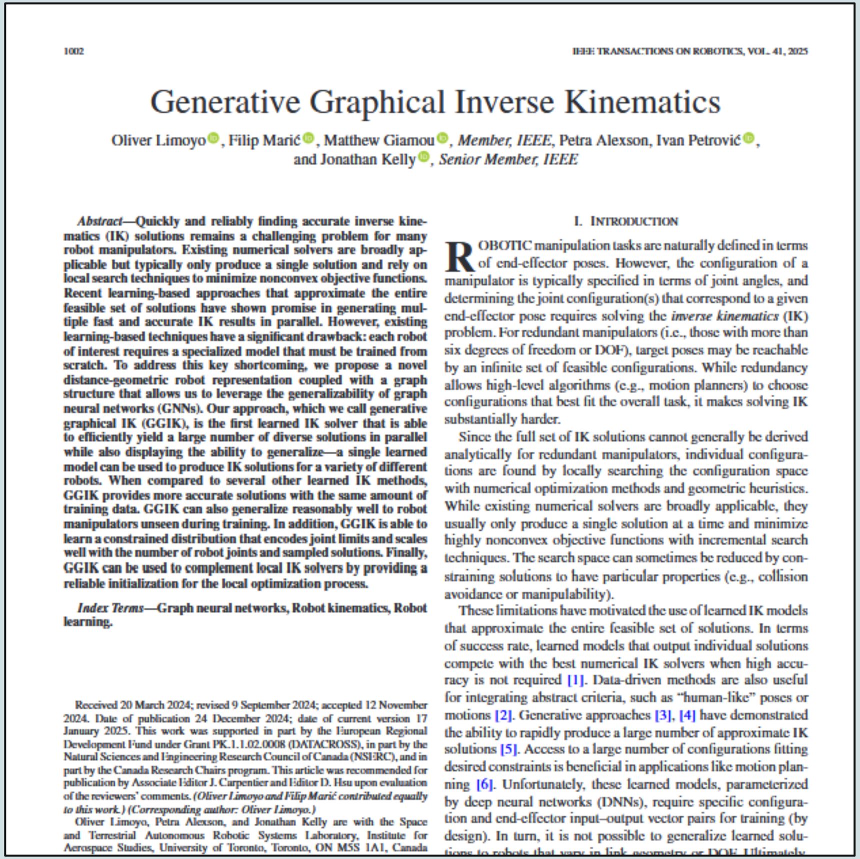
1. *Randomize joints to create a configuration*
2. *Use forward kinematics to calculate the end-effect position*



Generative Graphical Inverse Kinematics

Oliver Limoyo, Filip Marić, Matthew Giamou, Petra Alexson, Ivan Petrović, Jonathan Kelly

IEEE Transactions on Robotics, 2024



Generative Graphical Inverse Kinematics

Oliver Limoyo, Filip Marić, Matthew Giamou, *Member, IEEE*, Petra Alexson, Ivan Petrović, and Jonathan Kelly, *Senior Member, IEEE*

***Abstract*—Quickly and reliably finding accurate inverse kinematics (IK) solutions remains a challenging problem for many robot manipulators. Existing numerical solvers are broadly applicable but typically only produce a single solution and rely on local search techniques to minimize nonconvex objective functions. Recent learning-based approaches that approximate the entire feasible set of solutions have shown promise in generating multiple fast and accurate IK results in parallel. However, existing learning-based techniques have a significant drawback: each robot of interest requires a specialized model that must be trained from scratch. To address this key shortcoming, we propose a novel distance-geometric robot representation coupled with a graph structure that allows us to leverage the generalizability of graph neural networks (GNNs). Our approach, which we call generative graphical IK (GGIK), is the first learned IK solver that is able to efficiently yield a large number of diverse solutions in parallel while also displaying the ability to generalize—a single learned model can be used to produce IK solutions for a variety of different robots. When compared to several other learned IK methods, GGIK provides more accurate solutions with the same amount of training data. GGIK can also generalize reasonably well to robot manipulators unseen during training. In addition, GGIK is able to learn a constrained distribution that encodes joint limits and scales well with the number of robot joints and sampled solutions. Finally, GGIK can be used to complement local IK solvers by providing a reliable initialization for the local optimization process.**

***Index Terms*—Graph neural networks, Robot kinematics, Robot learning.**

Received 20 March 2024; revised 9 September 2024; accepted 12 November 2024. Date of publication 24 December 2024; date of current version 17 January 2025. This work was supported in part by the European Regional Development Fund under Grant PK.1.1.02.0008 (DATACROSS), in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Canada Research Chairs program. This article was recommended for publication by Associate Editor J. Carpentier and Editor D. Hsu upon evaluation of the reviewers' comments. *(Oliver Limoyo and Filip Marić contributed equally to this work.) (Corresponding author: Oliver Limoyo.)*

Oliver Limoyo, Petra Alexson, and Jonathan Kelly are with the Space and Terrestrial Autonomous Robotic Systems Laboratory, Institute for Aerospace Studies, University of Toronto, Toronto, ON M5S 1A1, Canada

I. Introduction

ROBOTIC manipulation tasks are naturally defined in terms of end-effector poses. However, the configuration of a manipulator is typically specified in terms of joint angles, and determining the joint configuration(s) that correspond to a given end-effector pose requires solving the *inverse kinematics* (IK) problem. For redundant manipulators (i.e., those with more than six degrees of freedom or DOF), target poses may be reachable by an infinite set of feasible configurations. While redundancy allows high-level algorithms (e.g., motion planners) to choose configurations that best fit the overall task, it makes solving IK substantially harder.

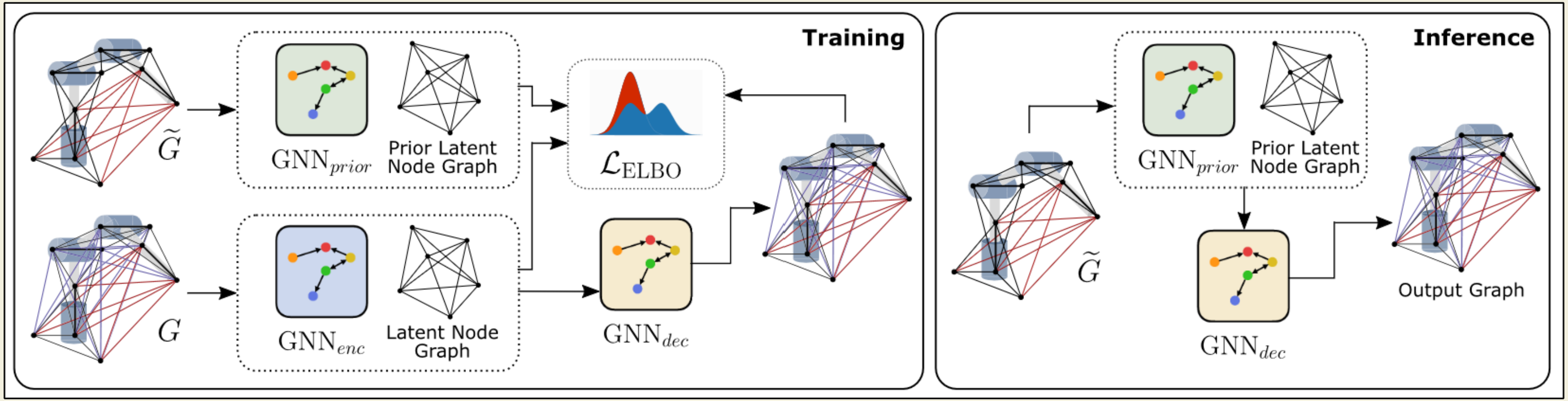
Since the full set of IK solutions cannot generally be derived analytically for redundant manipulators, individual configurations are found by locally searching the configuration space with numerical optimization methods and geometric heuristics. While existing numerical solvers are broadly applicable, they usually only produce a single solution at a time and minimize highly nonconvex objective functions with incremental search techniques. The search space can sometimes be reduced by constraining solutions to have particular properties (e.g., collision avoidance or manipulability).

These limitations have motivated the use of learned IK models that approximate the entire feasible set of solutions. In terms of success rate, learned models that output individual solutions compete with the best numerical IK solvers when high accuracy is not required [1]. Data-driven methods are also useful for integrating abstract criteria, such as "human-like" poses or motions [2]. Generative approaches [3], [4] have demonstrated the ability to rapidly produce a large number of approximate IK solutions [5]. Access to a large number of configurations fitting desired constraints is beneficial in applications like motion planning [6]. Unfortunately, these learned models, parameterized by deep neural networks (DNNs), require specific configuration and end-effector input–output vector pairs for training (by design). In turn, it is not possible to generalize learned solu-

2. GGIK Architecture

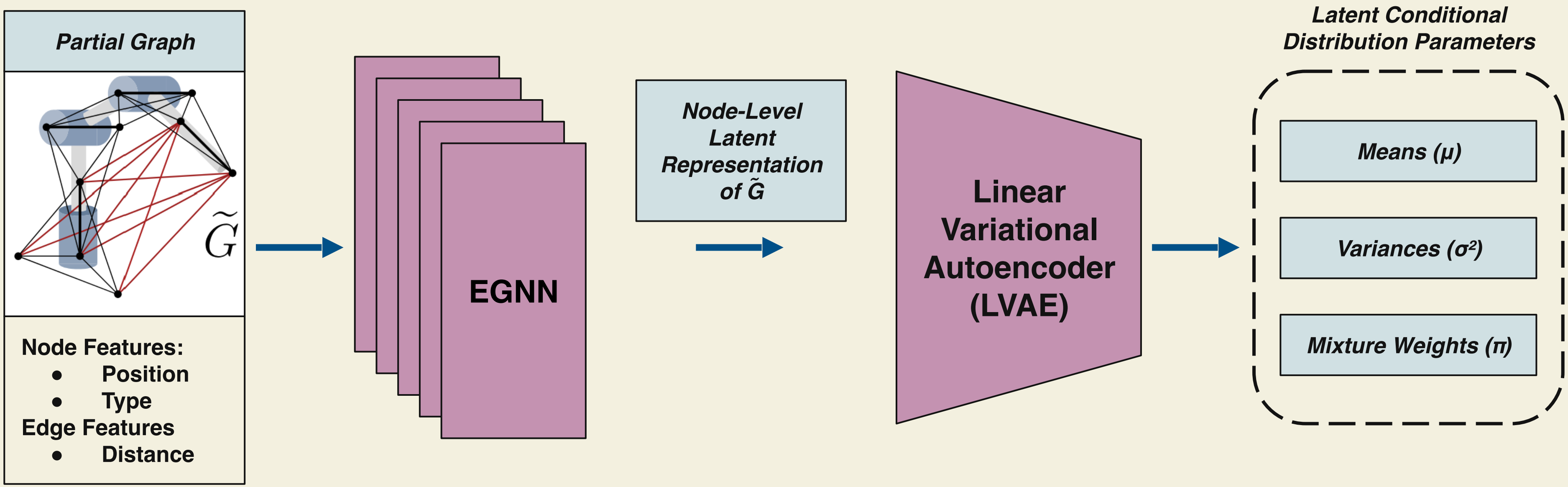
Architecture

GGIK is described as a ***Conditional Variational Autoencoder*** *(CVAE)*



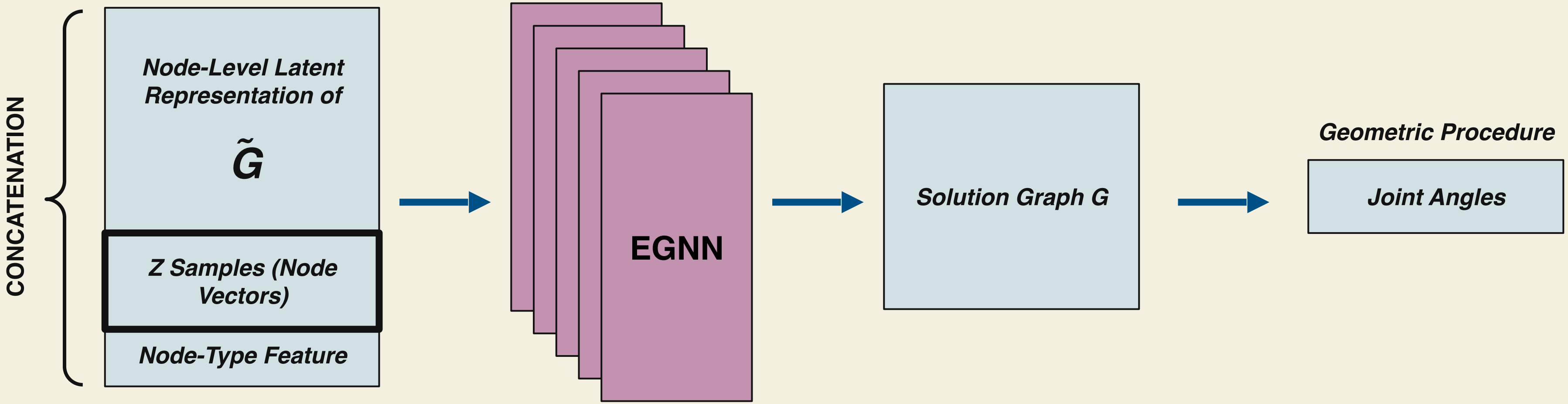
GNN$_{prior}$

Encodes a partial graph into latent space while optimizing the distribution, $p_{\gamma} (Z \mid \tilde{G})$



GNN$_{decoder}$

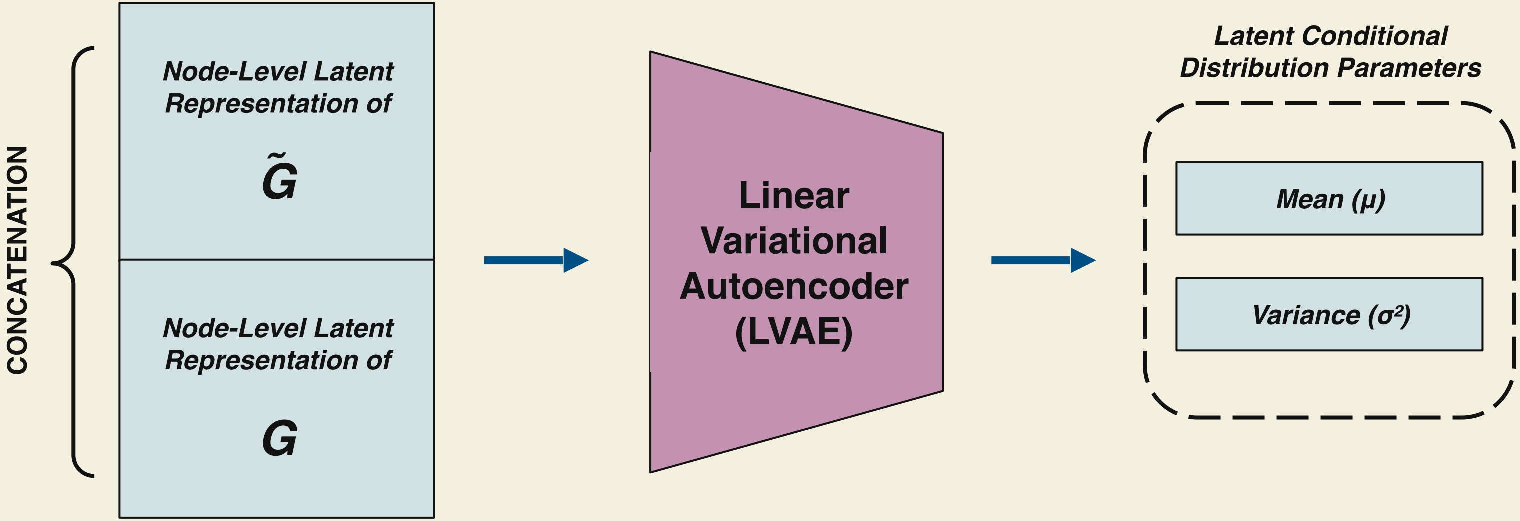
Decodes a sampled latent vector into G while optimizing the distribution, $p_\gamma(G \mid Z, \tilde{G})$



2. GGIK Architecture

GNN$_{encoder}$

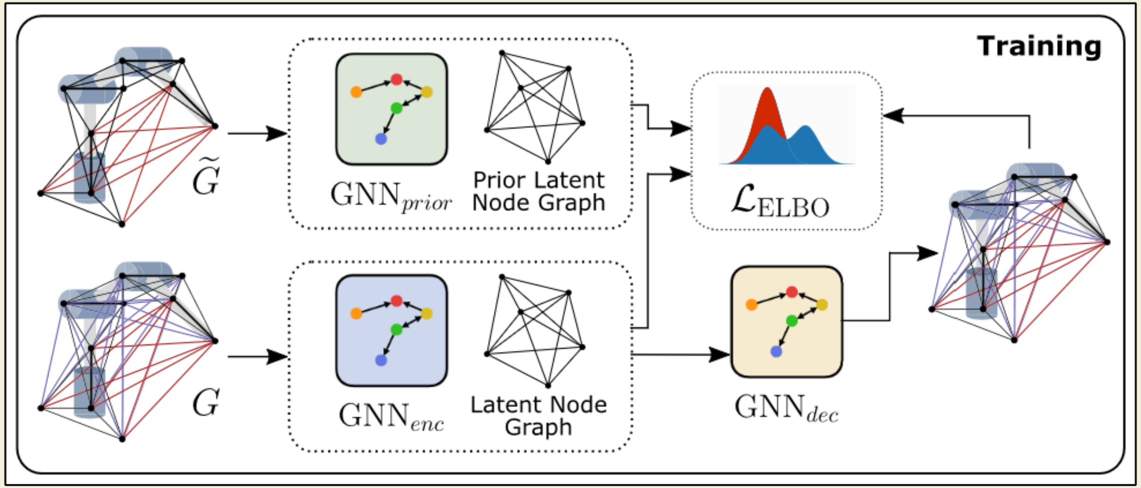
Encodes a complete graph into latent space while optimizing the distribution, $q_\phi(Z \mid G, \tilde{G})$



Architecture Revisited

GGIK optimizes three conditional latent distributions:

- $p_\gamma (Z \mid \tilde{G})$
- $p_\gamma (G \mid Z, \tilde{G})$
- $q_\phi (Z \mid G, \tilde{G})$



Optimization

*The **evidence lower bound** (ELBO) is used to optimize*

Reconstruction: How well a complete graph G matches itself after being encoded and decoded

KL Divergence: How close the latent distribution of a partial graph $\tilde{G}$ and complete graph G *are*

β-Scaling: Gradually increases the weight of the KL Divergence term

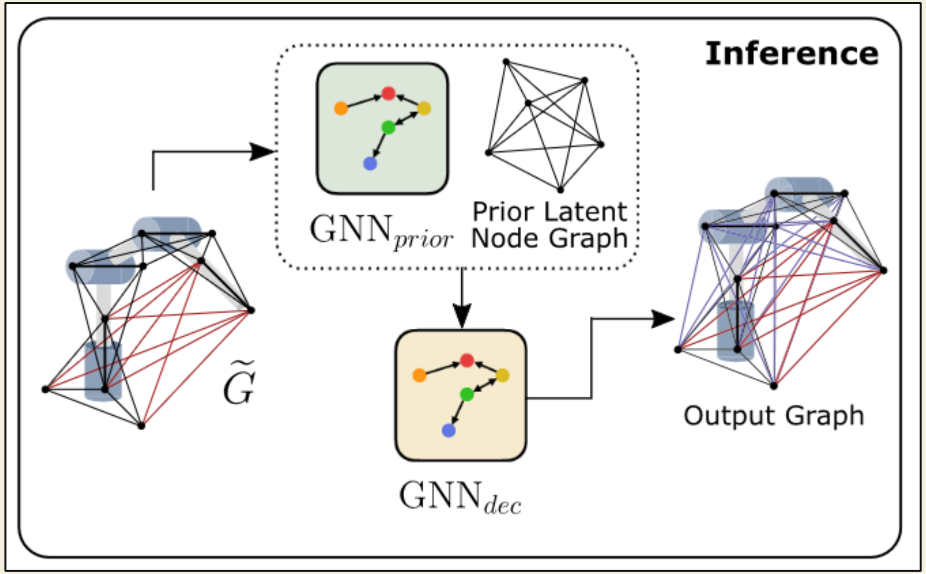
$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[\log p_\gamma(G \mid \widetilde{G}, \mathbf{Z})] - \beta \mathrm{KL}(q_\phi(\mathbf{Z} \mid G) \,||\, p_\gamma(\mathbf{Z} \mid \widetilde{G}))$$

Inference

At inference time, a partial graph can generate a distribution of complete graphs

*The specific solution is dependent on how the latent space **Z** is sampled*

- ***Z** is in the form of a Gaussian Mixture Model*



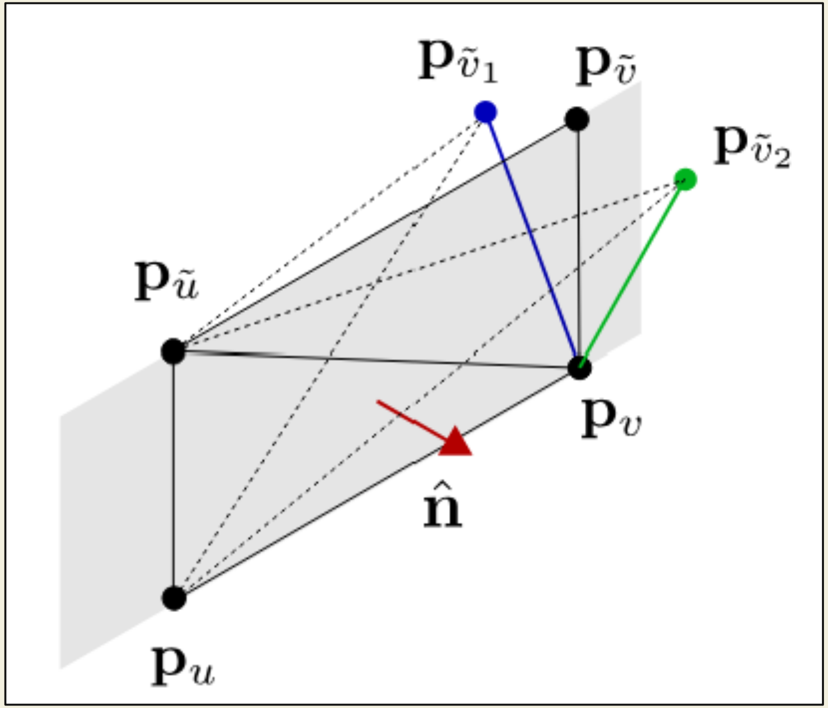
1. Problem Statement
2. GGIK Architecture
3. **Improvements**
4. Experimental Results
5. Drawing a Conclusion

The Issue of *Distance-Graphs*

When consecutive joints are not coplanar, we can no longer uniquely represent a robot

If $\mathbf{p}_{v1}$ replaces $\mathbf{p}_v$ and edge features include distance

Then the geometric graph is no longer unique

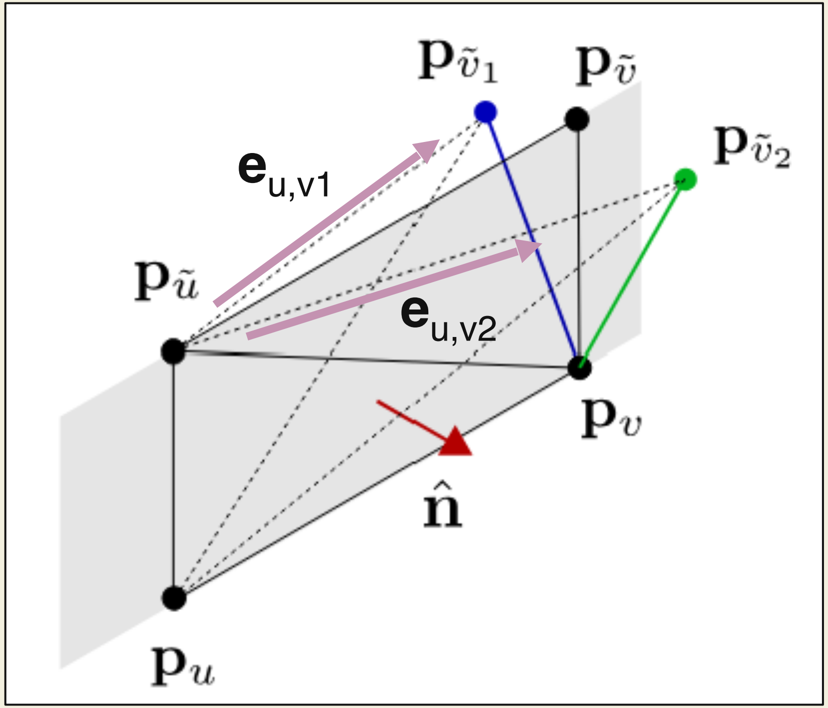


Proposal: *Distance-Direction-Graphs*

We propose to add direction to edge features

If $\mathbf{p}_{v1}$ replaces $\mathbf{p}_v$ and edge features include distance <u>and direction vectors</u>

Then the geometric graph <u>is unique</u>



Can we enhance GGIK to solve for non-coplanar robots and improve accuracy?

The problem

Determine whether attention over the direction of graph edges will improve accuracy on non-coplanar robots.

The objective

What Can We Improve Using Attention?

Directional Awareness

Learn how to weight messages based on **direction vectors,** capturing orientation sensitive relationships

Flexible Influence

Instead of treating all edges equally, learn **which neighbors are relevant**, both spatially and semantically

Better for Non-Coplanar

Helps focus updates on joints that are **aligned or structurally important**

Base Architecture/Changes Phi_E

Base Architecture

Purpose: Generates the raw message that passes through the GNN from one node to another

Inputs:

• h_i: Latent feature vector of receiving node. (node identity, info from neighbors, place in the structure)

• h_j: Latent feature vector of sending node.

• l lx_i - x_j l l² : Squared Euclidean distance .

• edge_attr: Scalar Euclidean distance (l lx_i - x_j l l) (1D)

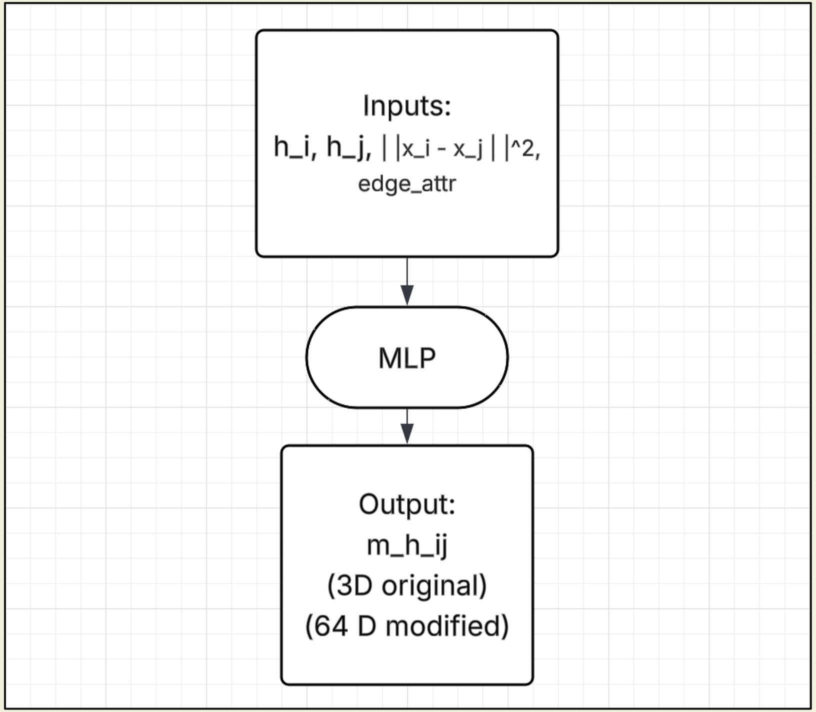
Output:

• m_h_ij : A learned abstract 3 dimensional vector that encodes semantic and relational information between the two nodes, one for each edge

Changes:

edge_attr: Scalar Euclidean distance (l lx_i - x_j l l) (1D) AND Directional Vector (x_j -x_i for (x,y,z))

m_h_ij : 64 dimensional vector



3. Improvements

Base Architecture/Changes Phi_X

Base Architecture

Purpose: Uses the message vector *m_h_ij* to compute a scalar used to scale the directional vector for updating node positions.

Input: m_h_ij : Message features computed by phi_e.
Output: m_x_ij = scalar × (x_j – x_i), equivariant directional update

Changes:

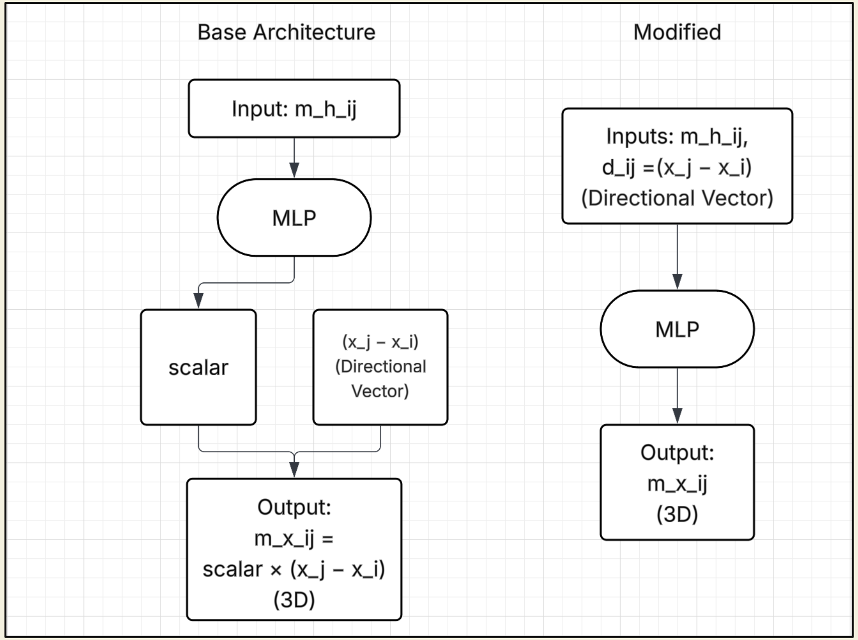
Purpose: Takes the message m_h_ij and the directional vector d_ij = x_j - x_i, and learns the geometric update to the node's position.

Inputs:

- m_h_ij : Message features computed by phi_e.
- d_ij: Directional Vector (x_j -x_i for (x,y,z))

Output: m_x_ij : still equivariant directional update, but now shaped by the joint interaction of the message and the vector geometry

Why?: It's crucial when the same distance can imply very different directional updates, which only happens in non-coplanar cases.



New! attn_mlp: Attention Scoring Network

Base Architecture

Purpose: Computes a scalar attention weight a_ij for each edge, based on the sending/receiving nodes and the edge attributes. This tells the GNN how important each message is.

Inputs:

• h_i: Latent feature vector of receiving node. (node identity, info from neighbors, place in the structure)

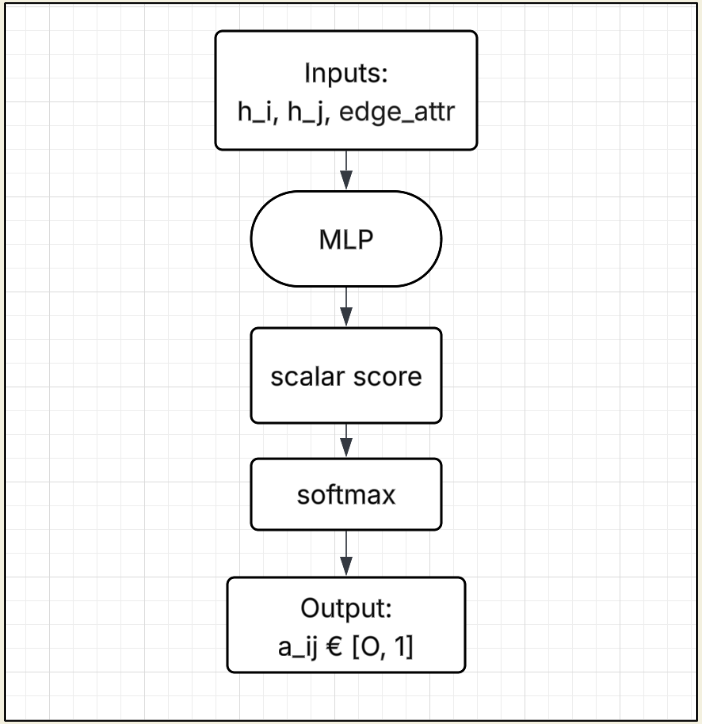
• h_j: Latent feature vector of sending node.

edge_attr: Scalar Euclidean distance (l lx_i - x_j l l) (1D) AND Directional Vector (x_j -x_i for (x,y,z))

Output:

• Scalar attention weights a_ij € [O, 1] normalized per target node via softmax.

• These are used to scale both m_h and m_x, changing how much influence each neighbor has.



Base Architecture/Changes Phi_H

Base Architecture

Purpose: Updates the node's latent embedding (h_i) by combining its current state with the aggregated messages (m_h) from its neighbors.

Inputs:

• h_i: Latent feature vector of receiving node. (node identity, info from neighbors, place in the structure)

• m_h: Aggregated message

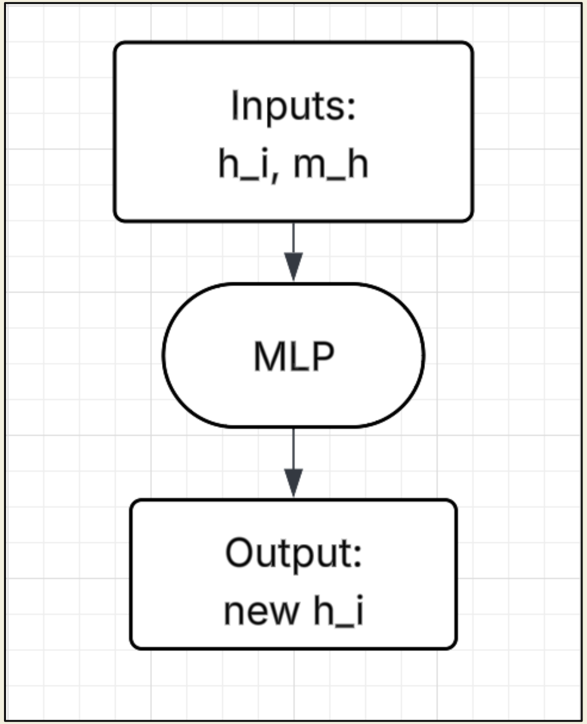
Output:

• new h_i: new latent feature vector, passed to the next GNN layer.

Changes:

m_h: 3D in original, 64 in modified, Also all node messages have same weight in aggregation in original, but are attention weighted in modified.

Why?: Neighbors now contribute based on learned relevance, and now have richer semantic and relational encoding



Flow Overview

For each edge (i <- j):

1. phi_e creates a message m_h_ij.

2. phi_x uses m_h_ij and direction d_ij to get geometric update m_x_ij.

3. attn_mip computes how much to weigh them (a_ij).

4. Both m_h_Ij and m_x_ij are scaled by a_ij.

5. The GNN aggregates these for each node. (Σ_j α_ij m_x_ij and Σ_j α_ij m_h_ij) to get m_x and m_h

6. phi_h fuses h_i with aggregated m_h to update h_i.

7. The node's position x_i is updated via x_i' = x_i + m_x / c (normalized if needed).

Important functions:

forward(): receives node positions x, hidden features h, edge attributes, and edge indices. Calls **propagate(),** kicking off the message passing. Same in both

propagate(): PyTorch Geometric internal dispatcher. It delegates to **message(), aggregate()**, and **update()**. Same in both

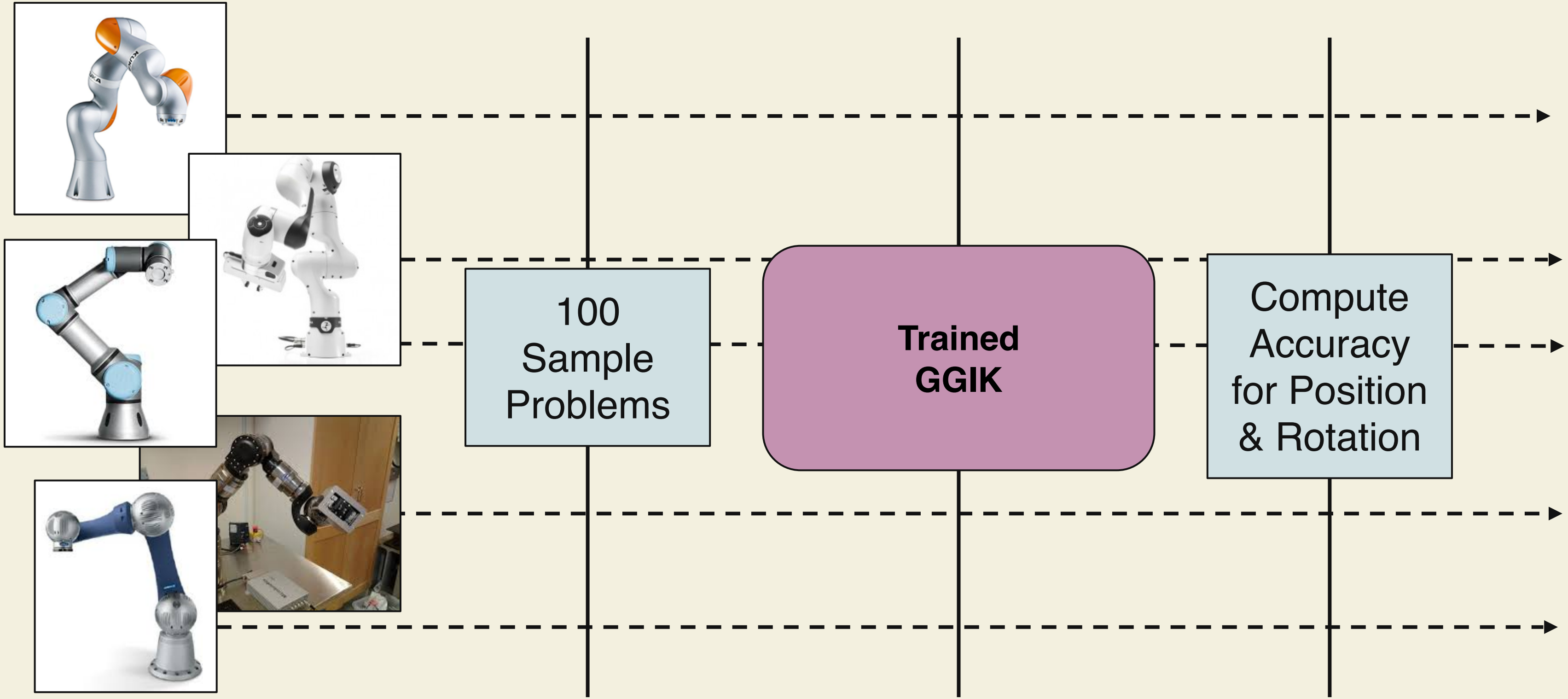
message(): Computes messages from each neighbor j → i. Uses phi_e, phi_x and attention_mlp which are different in modified version. Scales messages with attention in modified.

aggregate(): Collects all messages per node. Attention weighted messages in modified version.

update(): calls phi_h to update h_i and does x_i + m_x / c to update x_i

1. Problem Statement
2. GGIK Architecture
3. Improvements
4. **Experimental Results**
5. Drawing a Conclusion

Experimental Procedure



GGIK vs. GGIK+ Training

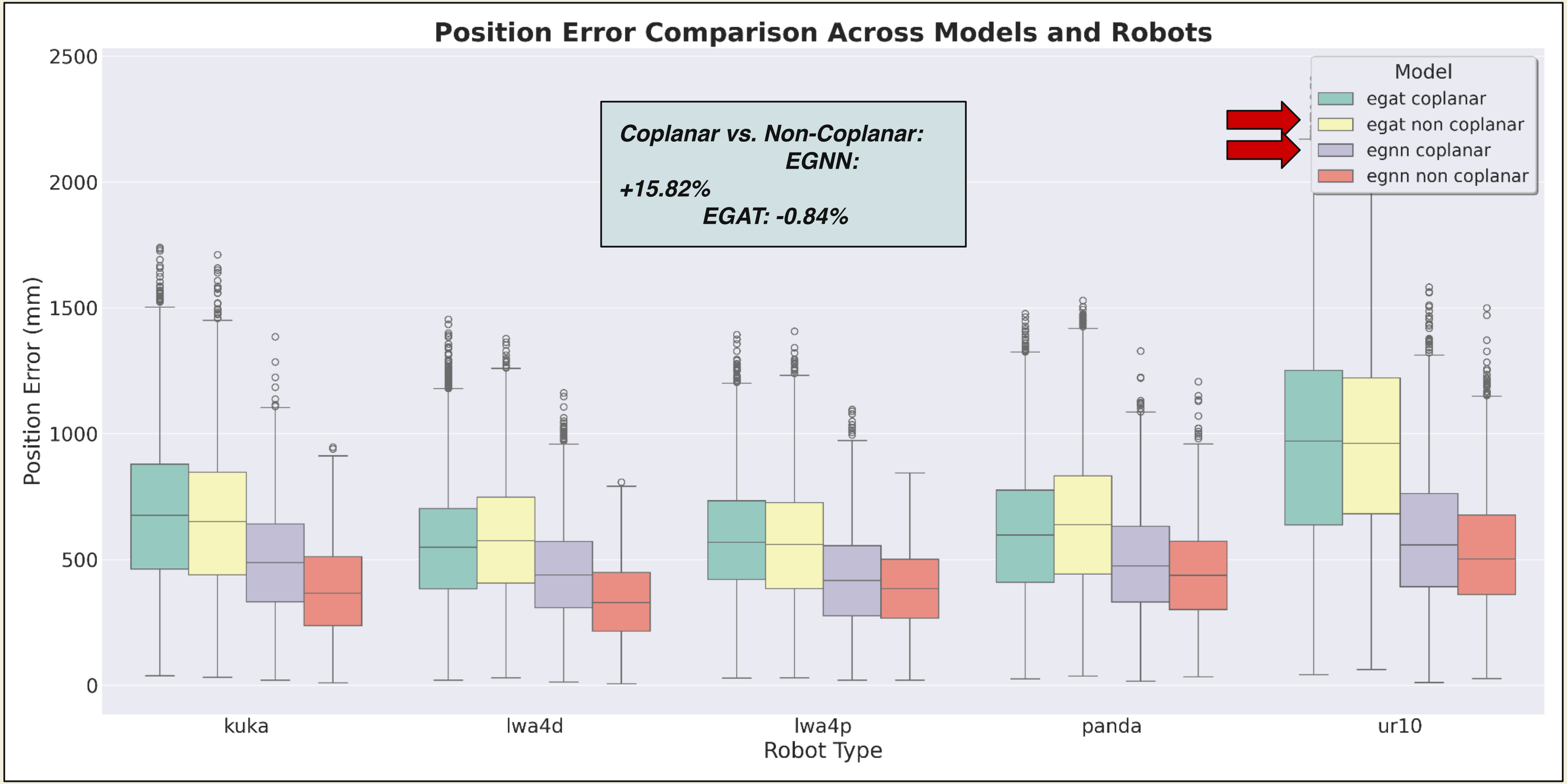
	GGIK (GNN)	**GGIK (GAT)**
Samples	4,096,000	5,120
Epochs	300	50

How does EGAT compare to EGNN?

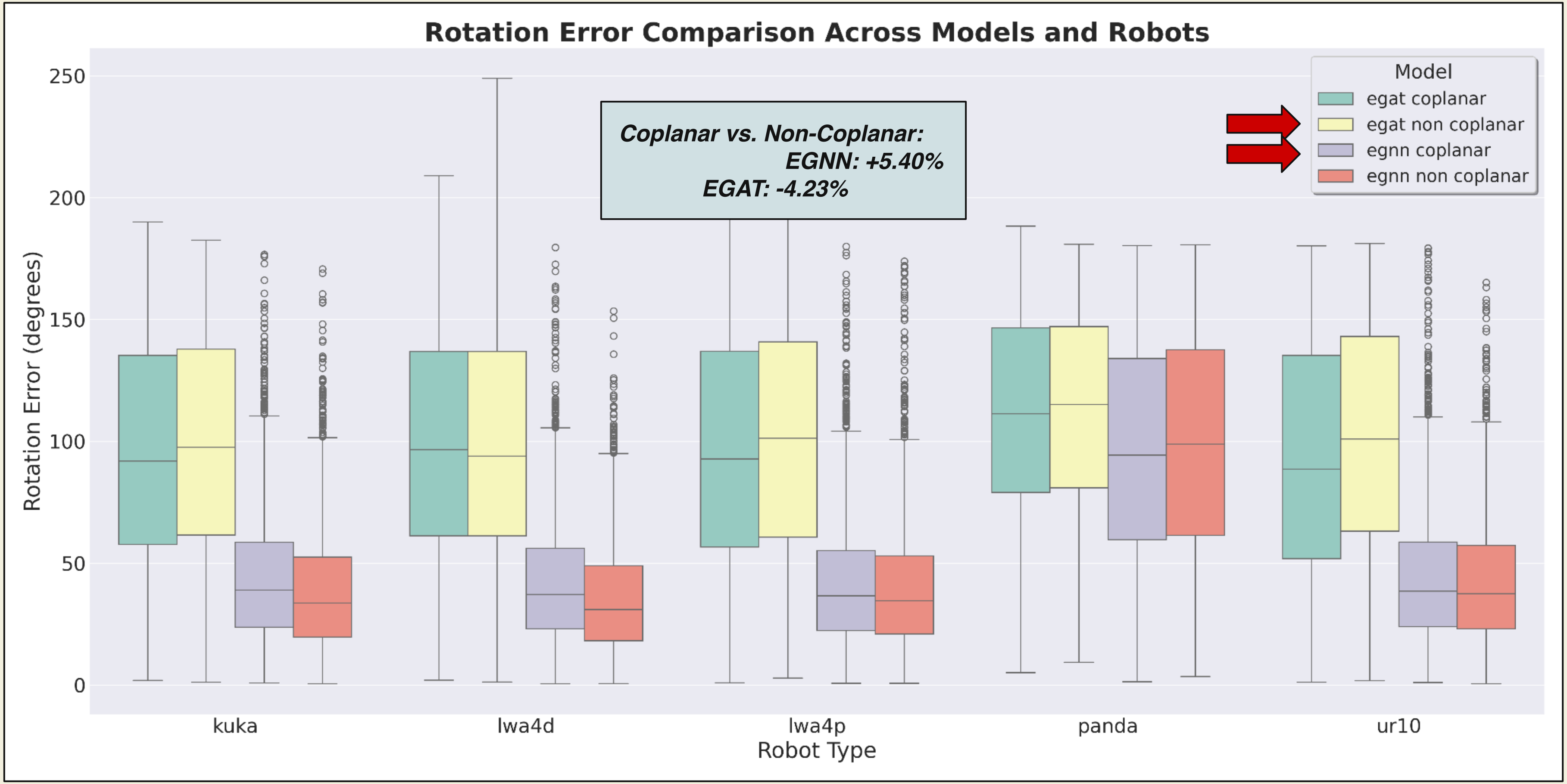
Not Good!

-34.19%

*But with the **ability to the solve for non-coplanar robots!***

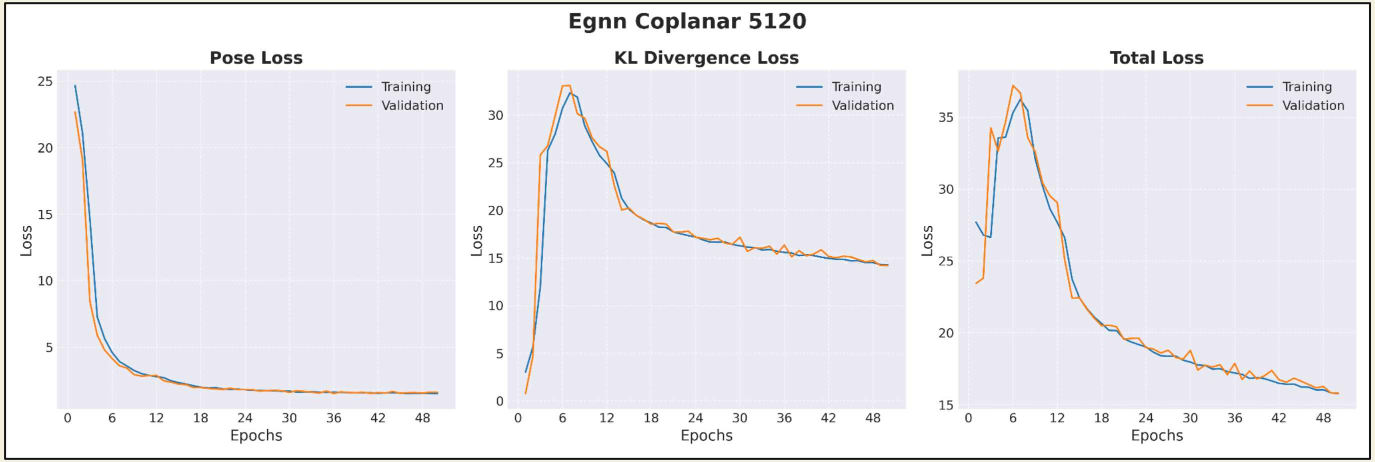


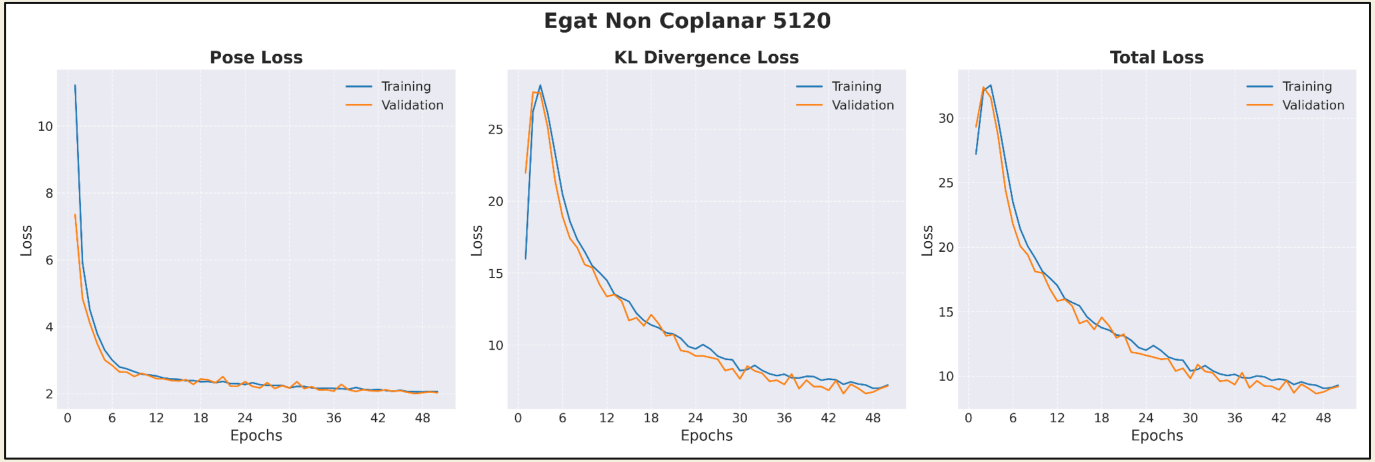
4. Experimental Results



4. Experimental Results

EGNN EGAT: Loss Curves





1. Problem Statement
2. GGIK Architecture
3. Improvements
4. Experimental Results
5. **Drawing a Conclusion**

Key Findings

EGNN Handles Non-Coplanar Data

- When training on non-coplanar data vs. coplanar data:
    - EGNN: +15.82%
    - EGAT: -0.84%

EGAT Reaches a Lower KL Divergence Loss

- Attention is likely helping the model learn a **better latent representation**

- Attention may be **hurting the the decoding process**

E(n) Equivariant Graph Neural Networks

Victor Garcia Satorras, Emiel Hoogeboom, Max Welling

arXiv.org, 2022

E(n) Equivariant Graph Neural Networks

Victor Garcia Satorras [1] Emiel Hoogeboom [1] Max Welling [1]

Abstract

This paper introduces a new model to learn graph neural networks equivariant to rotations, translations, reflections and permutations called E(n)-Equivariant Graph Neural Networks (EGNNs). In contrast with existing methods, our work does not require computationally expensive higher-order representations in intermediate layers while it still achieves competitive or better performance. In addition, whereas existing methods are limited to equivariance on 3 dimensional spaces, our model is easily scaled to higher-dimensional spaces. We demonstrate the effectiveness of our method on dynamical systems modelling, representation learning in graph autoencoders and predicting molecular properties.

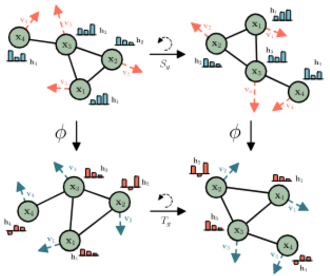


Figure 1. Example of rotation equivariance on a graph with a graph neural network ϕ

1. Introduction

Although deep learning has largely replaced hand-crafted features, many advances are critically dependent on inductive biases in deep neural networks. An effective method to restrict neural networks to relevant functions is to exploit the *symmetry* of problems by enforcing equivariance with respect to transformations from a certain symmetry group. Notable examples are translation equivariance in Convolutional Neural Networks and permutation equivariance in Graph Neural Networks (Bruna et al., 2013; Defferrard et al., 2016; Kipf & Welling, 2016a).

Many problems exhibit 3D translation and rotation symmetries. Some examples are point clouds (Uy et al., 2019), 3D molecular structures (Ramakrishnan et al., 2014) or N-body particle simulations (Kipf et al., 2018). The group corresponding to these symmetries is named the Euclidean group: SE(3) or when reflections are included E(3). It is often desired that predictions on these tasks are either equivariant or

Recently, various forms and methods to achieve E(3) or SE(3) equivariance have been proposed (Thomas et al., 2018; Fuchs et al., 2020; Finzi et al., 2020; Köhler et al., 2020). Many of these works achieve innovations in studying types of higher-order representations for intermediate network layers. However, the transformations for these higher-order representations require coefficients or approximations that can be expensive to compute. Additionally, in practice for many types of data the inputs and outputs are restricted to scalar values (for instance temperature or energy, referred to as type-0 in literature) and 3d vectors (for instance velocity or momentum, referred to as type-1 in literature).

In this work we present a new architecture that is translation, rotation and reflection equivariant (E(n)), and permutation

The Importance of E(n) Equivariance

Without EGNN this model would fail dramatically

In an equivariant system, a translation of an input will produce the identical output translated

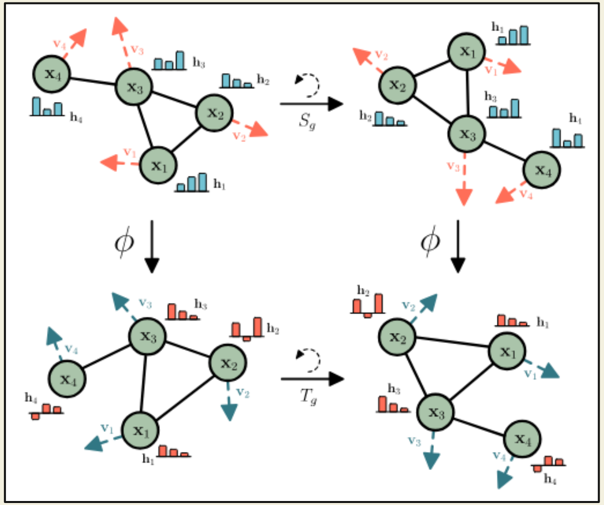


TABLE V
COMPARISON OF DIFFERENT NETWORK ARCHITECTURES

Model Name	Err. Pos. [mm]					Err. Rot. [deg]					Test ELBO
	mean	min	max	Q_1	Q_3	mean	min	max	Q_1	Q_3	
EGNN [59]	4.6	1.5	8.5	3.3	5.8	0.4	0.1	0.6	0.3	0.4	−0.05
MPNN [64]	143.2	62.9	273.7	113.1	169.1	17.7	5.3	13.6	21.6	34.1	−8.3
GAT [65]	-	-	-	-	-	-	-	-	-	-	−12.41
GCN [66]	-	-	-	-	-	-	-	-	-	-	−12.42
GRAPHsage [67]	-	-	-	-	-	-	-	-	-	-	−10.5

Future Work

Concatenate Layer Types

GAT, GCN, GNN layers perform worse, but may still have some value

Concatenate EGNN+GAT/GCN/GNN

Biased Data Generation

GGIK has been proven to respond positively to biased data

Bias data towards more stable solutions

Cascaded CVAE For High Precision

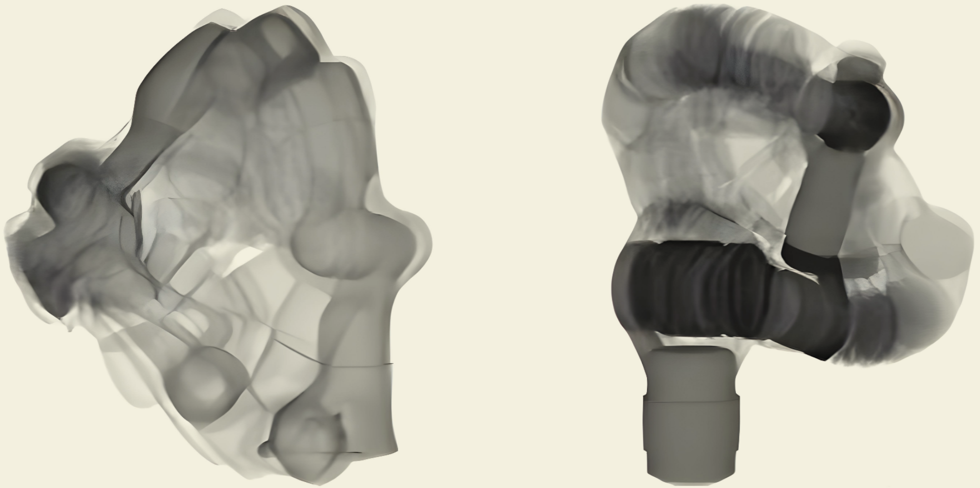
Take inspiration from Cascading “Super-Resolution” Diffusion models

A cascaded GGIK model can use the previous latent space as a starting point

- Relaxed Coplanar Restrictions
- Quantified Non-Coplanar Dataset
- Inserted Attention
- Quantified Attention-Enhanced Model

Thank you!

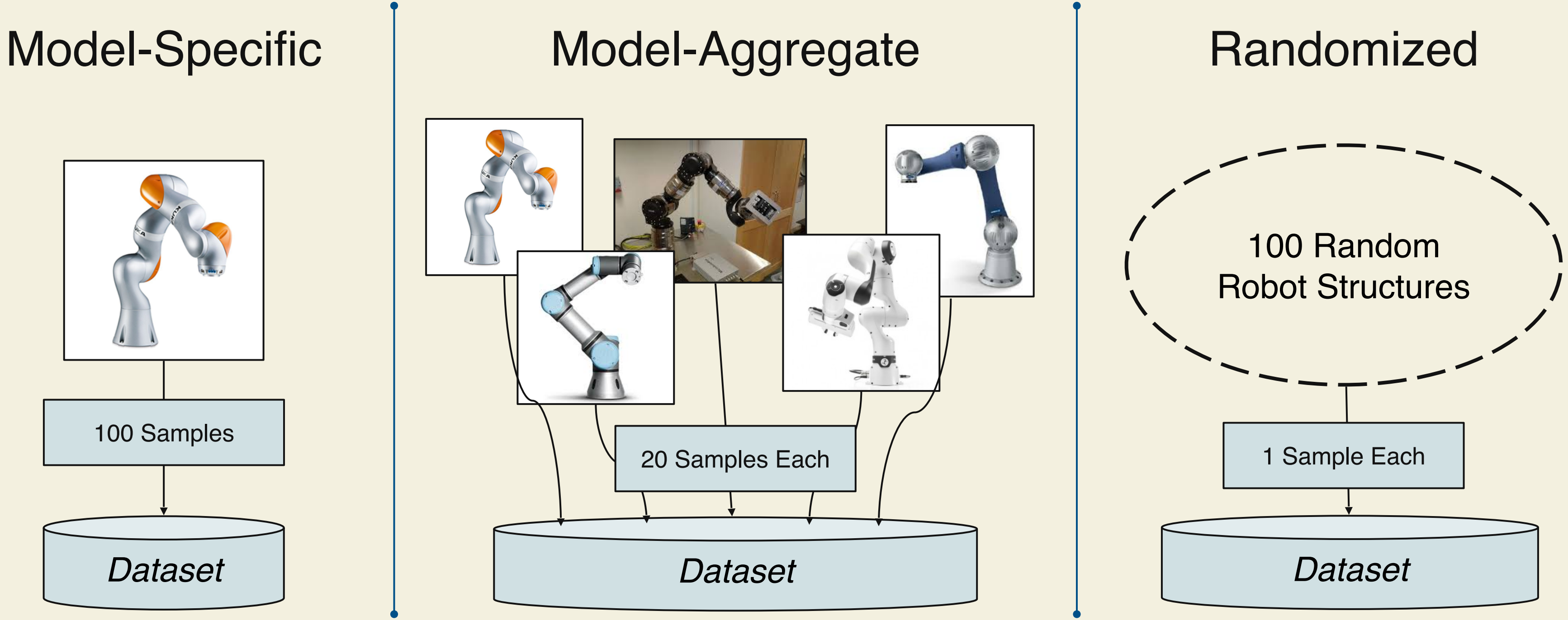
Calvin Stahoviak
Gabriel Urbaitis



Institution: *University of New Mexico*
Course: *ST, Advance Machine Learning*
Professor: *Trilce Estrada*

Supplemental Slides

Types of Datasets



VAE vs. CVAE

A variational autoencoder learns a normal distribution

$$p(G) = \int p(G \mid Z)\, p(Z)\, dZ$$

A conditional variational autoencoder learns a distribution conditioned on another variable

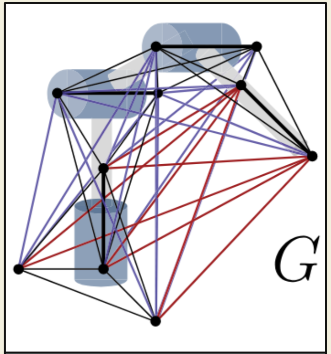
$$p(G \mid \tilde{G}) = \int p(G \mid Z, \tilde{G})\, p(Z, \tilde{G})\, dZ$$

Each output is sampled given its input

Distance Geometry Problem (DGP)

DGP asks, what is the location of a set of points given only the edge distances between them?

GGIK Output



Set of Edges

DGP Solver

Set of Node Points

Set of Edges

Geometric Procedure

Joint Angles