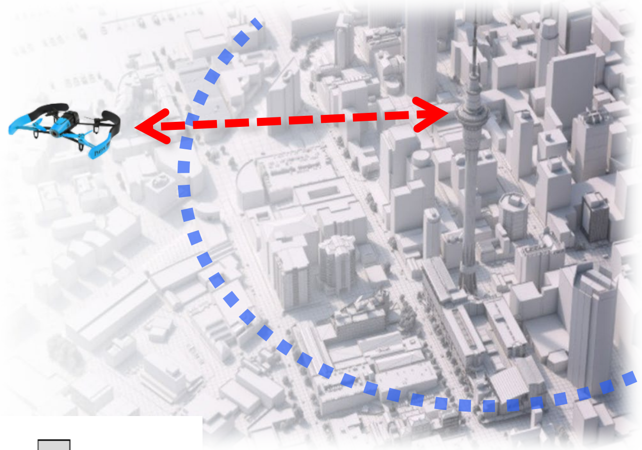
DEEP LEARNING AND GRAPH NEURAL NETWORKS FOR AUTONOMY

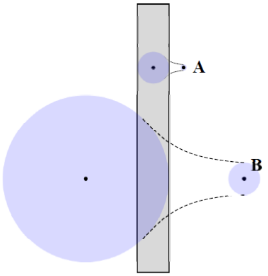
DR. WARREN E. DIXON

Department of Mechanical and Aerospace Engineering,
University of Florida

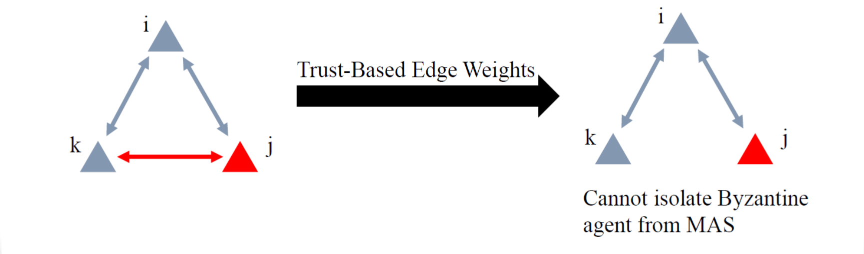
April 7th, 2025

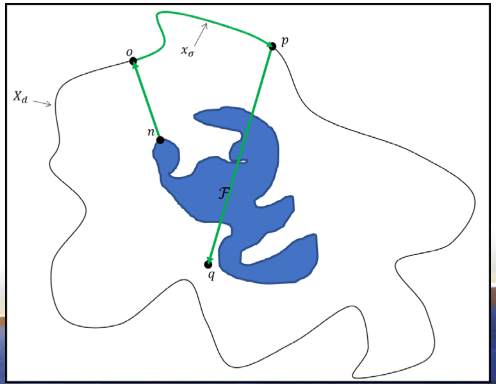
- Causes of temporary feedback loss
  - Task definition
    - Communication restricted operations
  - Operating environment
    - Intermittent occlusions of sensor signals
    - GPS denied regions
  - Sensor modality
    - Limited camera field-of-view
  - Cyber Effects

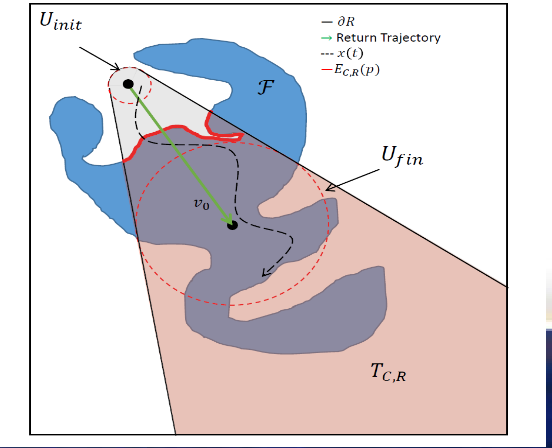




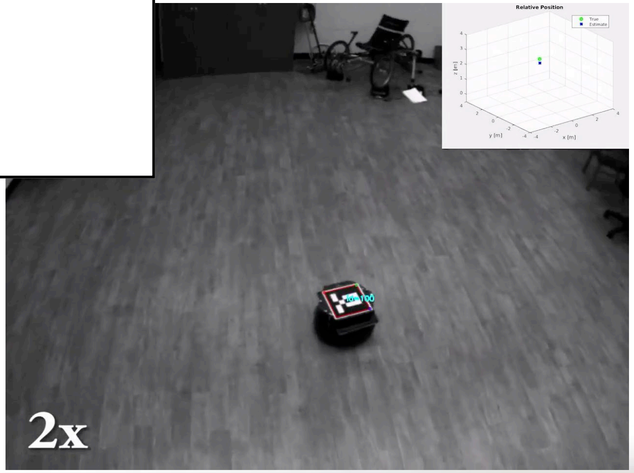
Topological Transition Guarantee

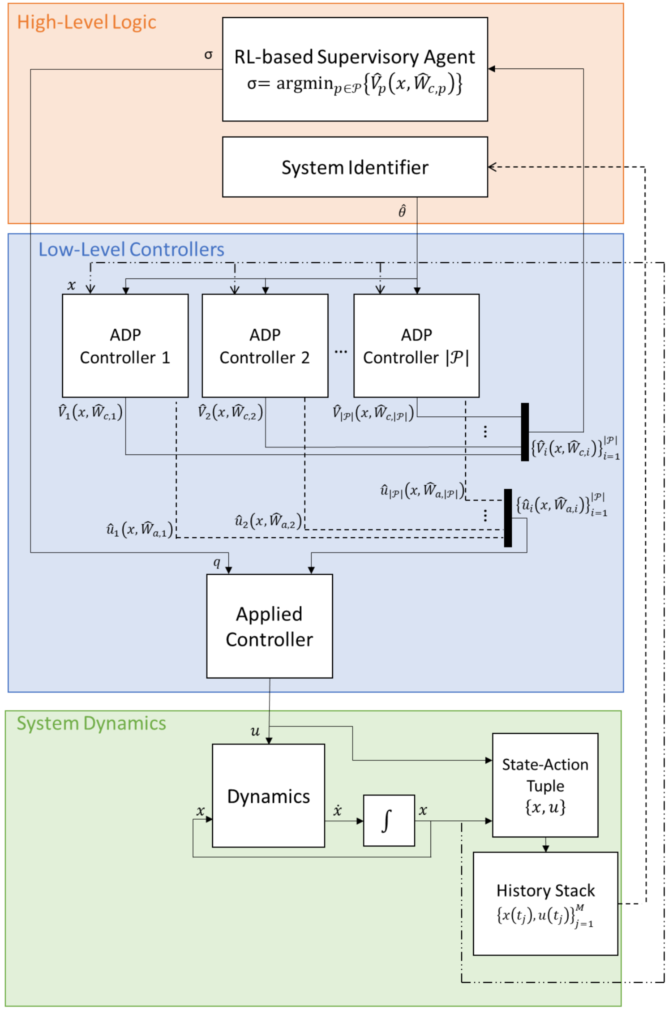


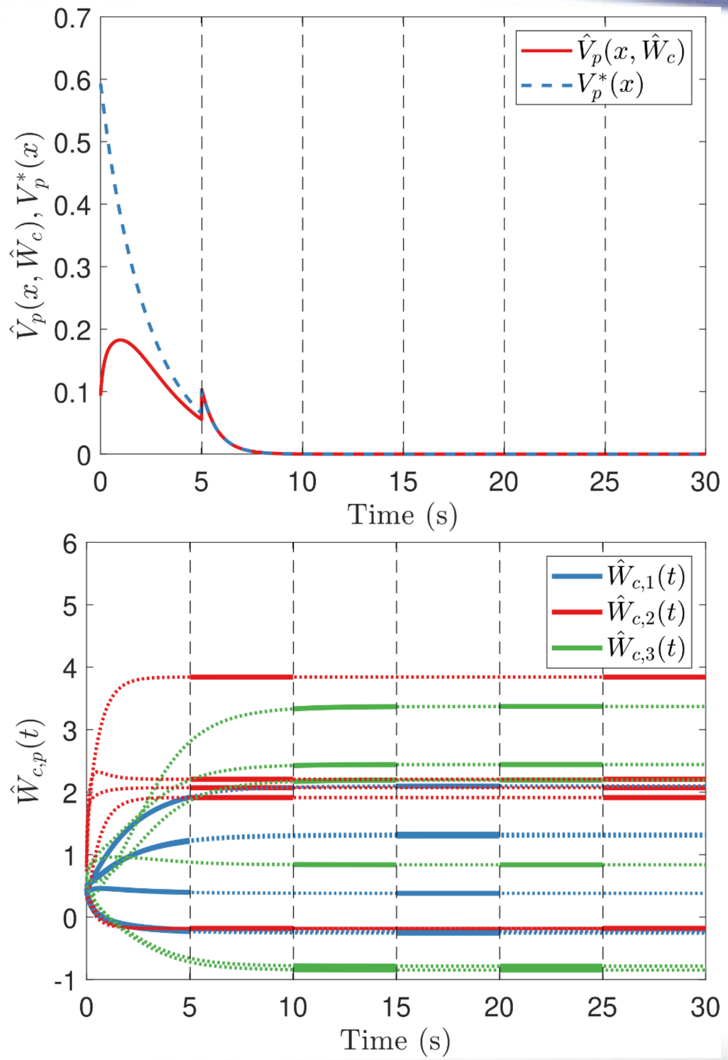


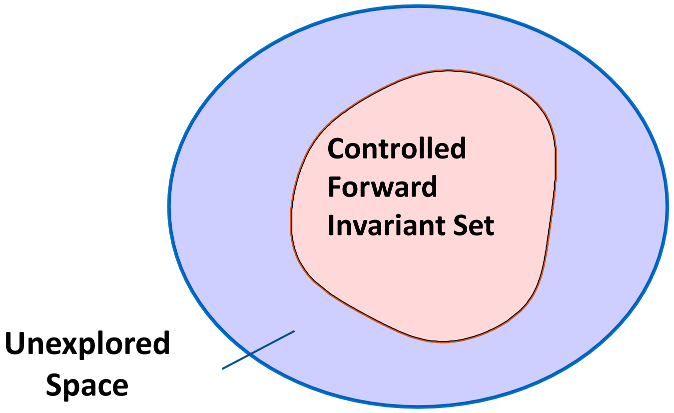


RELAY-EXPLORER PROBLEMS

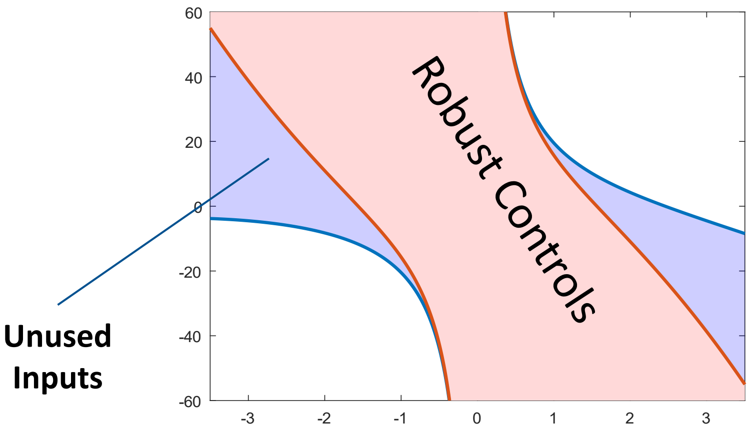


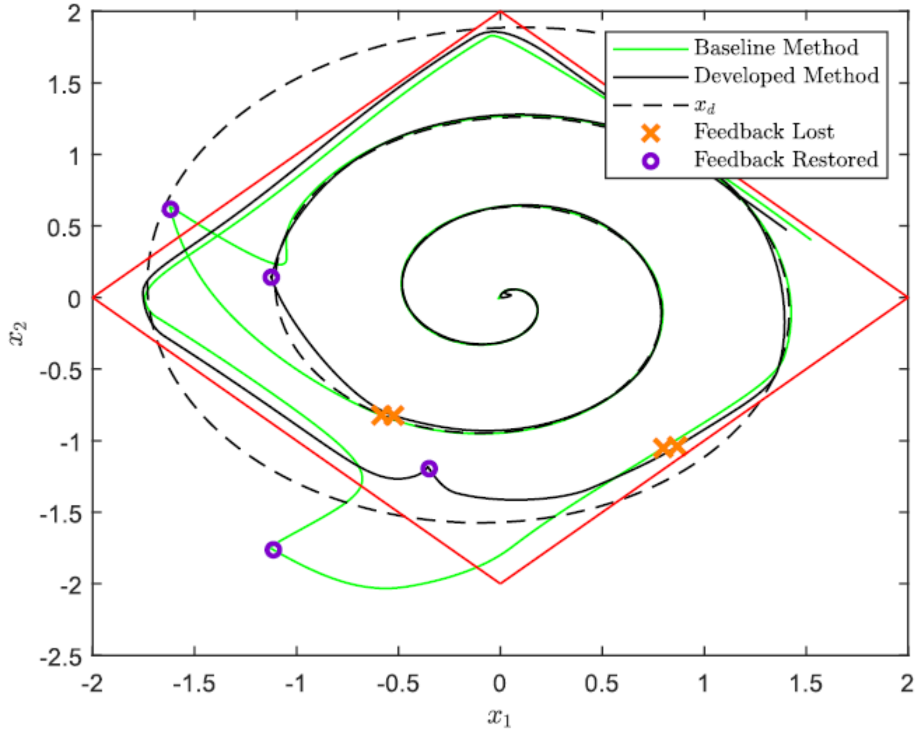

High-Level Logic
RL-based Supervisory Agent
σ= $\mathrm{argmin}_{p\in\mathcal{P}}\{\hat{V}_p(x,\hat{W}_{c,p})\}$
σ
System Identifier
$\hat{\theta}$
Low-Level Controllers
x
ADP Controller 1
ADP Controller 2
ADP Controller $|\mathcal{P}|$
$\hat{V}_1(x,\hat{W}_{c,1})$
$\hat{V}_2(x,\hat{W}_{c,2})$
$\hat{V}_{|\mathcal{P}|}(x,\hat{W}_{c,|\mathcal{P}|})$
$\{\hat{V}_i(x,\hat{W}_{c,i})\}_{i=1}^{|\mathcal{P}|}$
$\hat{u}_{|\mathcal{P}|}(x,\hat{W}_{a,|\mathcal{P}|})$
$\{\hat{u}_i(x,\hat{W}_{a,i})\}_{i=1}^{|\mathcal{P}|}$
$\hat{u}_1(x,\hat{W}_{a,1})$
$\hat{u}_2(x,\hat{W}_{a,2})$
q
Applied Controller
System Dynamics
u
Dynamics
$\dot{x}$
$\int$
x
State-Action Tuple $\{x,u\}$
History Stack $\{x(t_j),u(t_j)\}_{j=1}^{M}$


$\hat{V}_p(x,\hat{W}_c), V_p^*(x)$
$\hat{V}_p(x,\hat{W}_c)$
$V_p^*(x)$
Time (s)
$\hat{W}_{c,p}(t)$
$\hat{W}_{c,1}(t)$
$\hat{W}_{c,2}(t)$
$\hat{W}_{c,3}(t)$
Time (s)



Admissible Control Inputs





WHY DEEP LEARNING?

WHY DEEP LEARNING?

UNCERTAINTY: DEEP LEARNING

- Deep Learning is a machine learning method that has shown significant advances in pattern matching tasks – but not well suited for feedback control
  - Requires massive amounts of training data
  - Significant training time
  - Closed training sets, with no guarantees of convergence or stability
  - Implemented in open-loop – no online adaptation

- We recently developed a series of Deep Learning methods that can be applied in real-time, with no prior data, no training phase, with feedback-based (continuous) learning
  - …but more data and training is better
  - Stability analysis derived adaptation laws (with proof of convergence)
  - Assured Learning

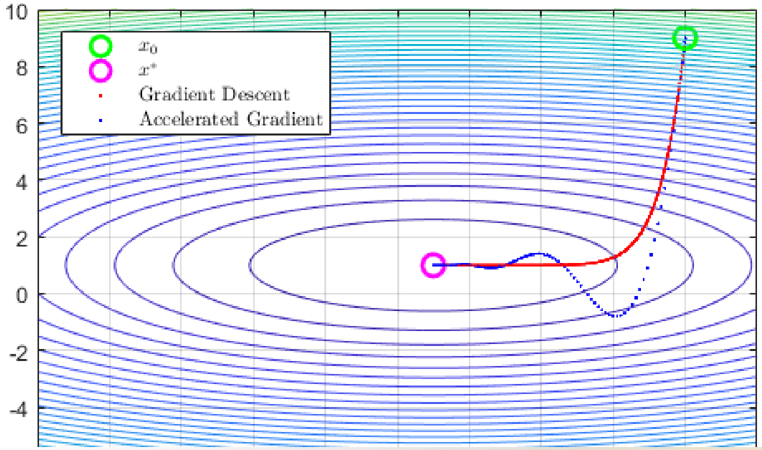
- Fully-Connected DNN with some input η

$$\Phi(\eta, V_0, V_1, \ldots, V_k) \triangleq \left(V_k^T \phi_k \circ \ldots \circ V_1^T \phi_1\right) \left(V_0^T \eta\right)$$

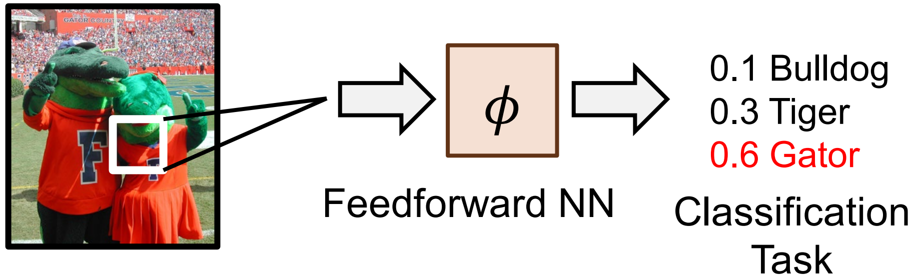
- Recursive Representation

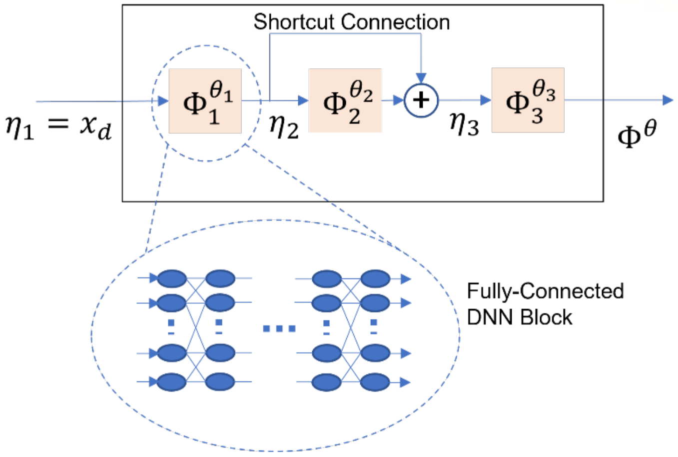
$$\Phi = V_k^T \varphi_k \qquad \varphi_j \triangleq \begin{cases} \phi_j \left(V_{j-1}^T \varphi_{j-1}\right), & j \in \{1, \ldots, k\}, \\ \eta, & j = 0. \end{cases}$$

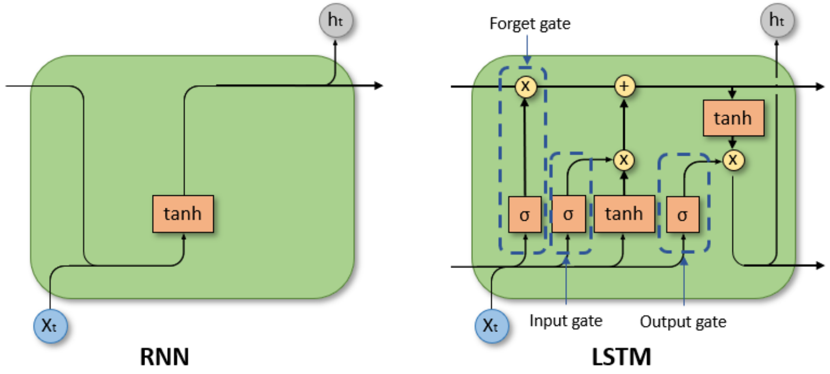
Accelerated Gradient

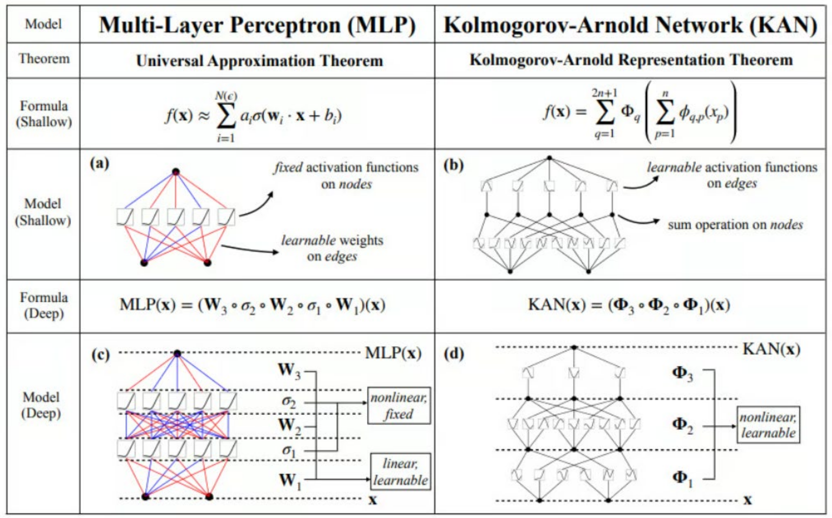


Convolutional neural network (CNN)

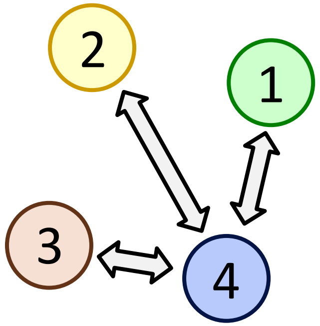








Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) *fixed* activation functions on *nodes*; *learnable* weights on *edges*	(b) *learnable* activation functions on *edges*; sum operation on *nodes*
Formula (Deep)	$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$
Model (Deep)	(c) MLP(x); $\mathbf{W}_3$; σ_2; $\mathbf{W}_2$; σ_1; $\mathbf{W}_1$; *nonlinear, fixed*; *linear, learnable*; x	(d) KAN(x); $\mathbf{\Phi}_3$; $\mathbf{\Phi}_2$; $\mathbf{\Phi}_1$; *nonlinear, learnable*; x



Graph neural network (GNN)

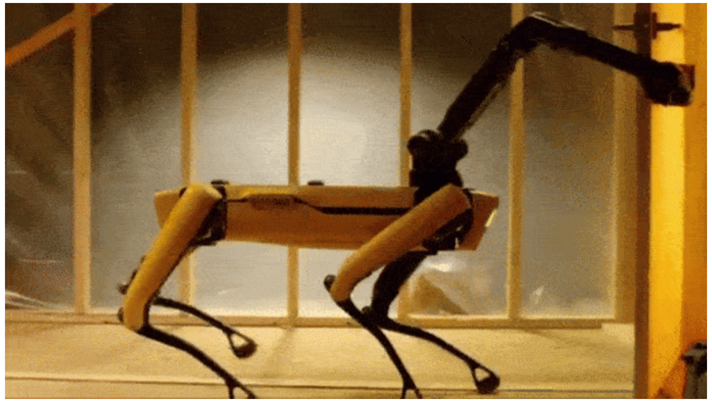
LYAPUNOV-BASED GRAPH NEURAL NETWORKS FOR MULTI-AGENT ADAPTIVE CONTROL

BRANDON C. FALLIN, CRISTIAN F. NINO, OMKAR SUDHIR PATIL, AND WARREN E. DIXON

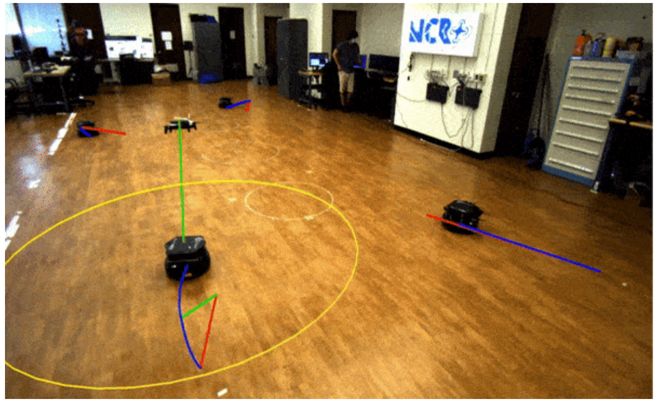
Department of Mechanical and Aerospace Engineering,
University of Florida

April 7th, 2025

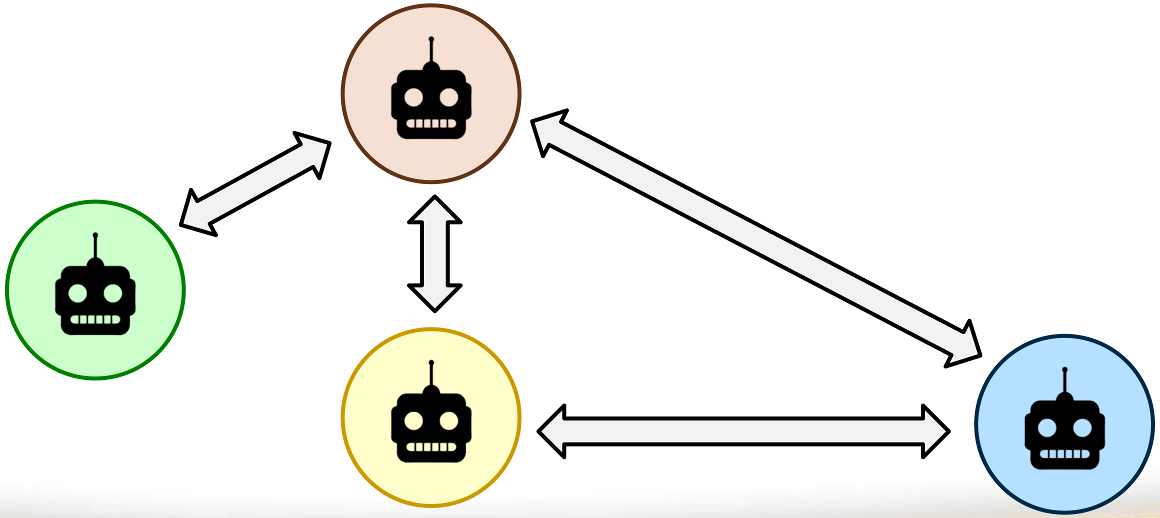
Credit: FAST Lab



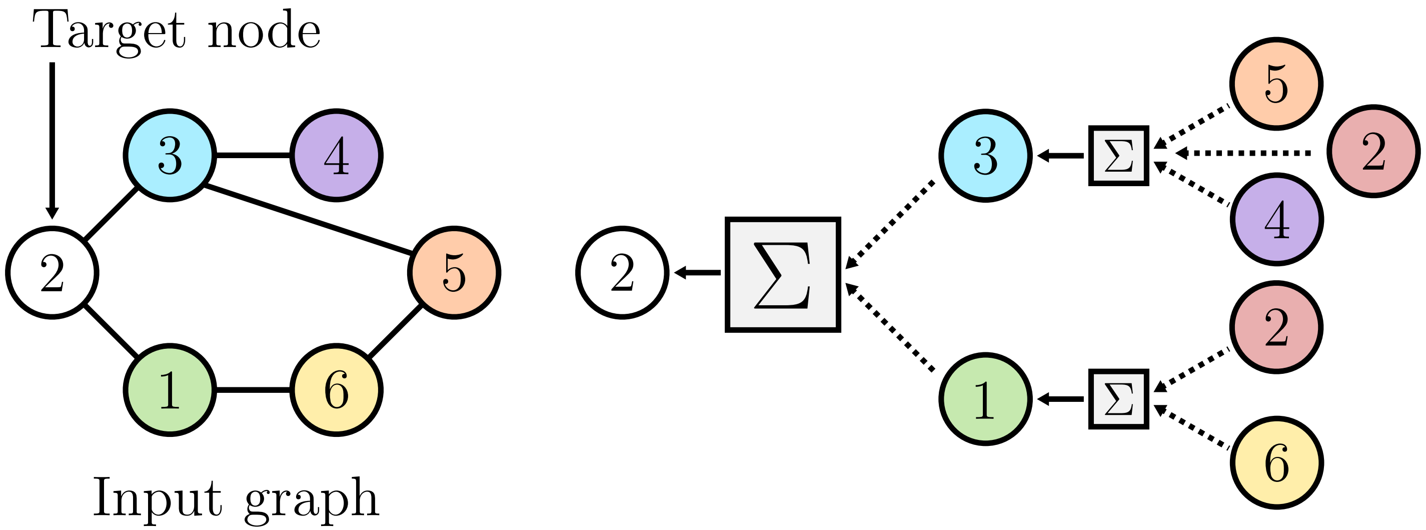
Credit: Boston Dynamics

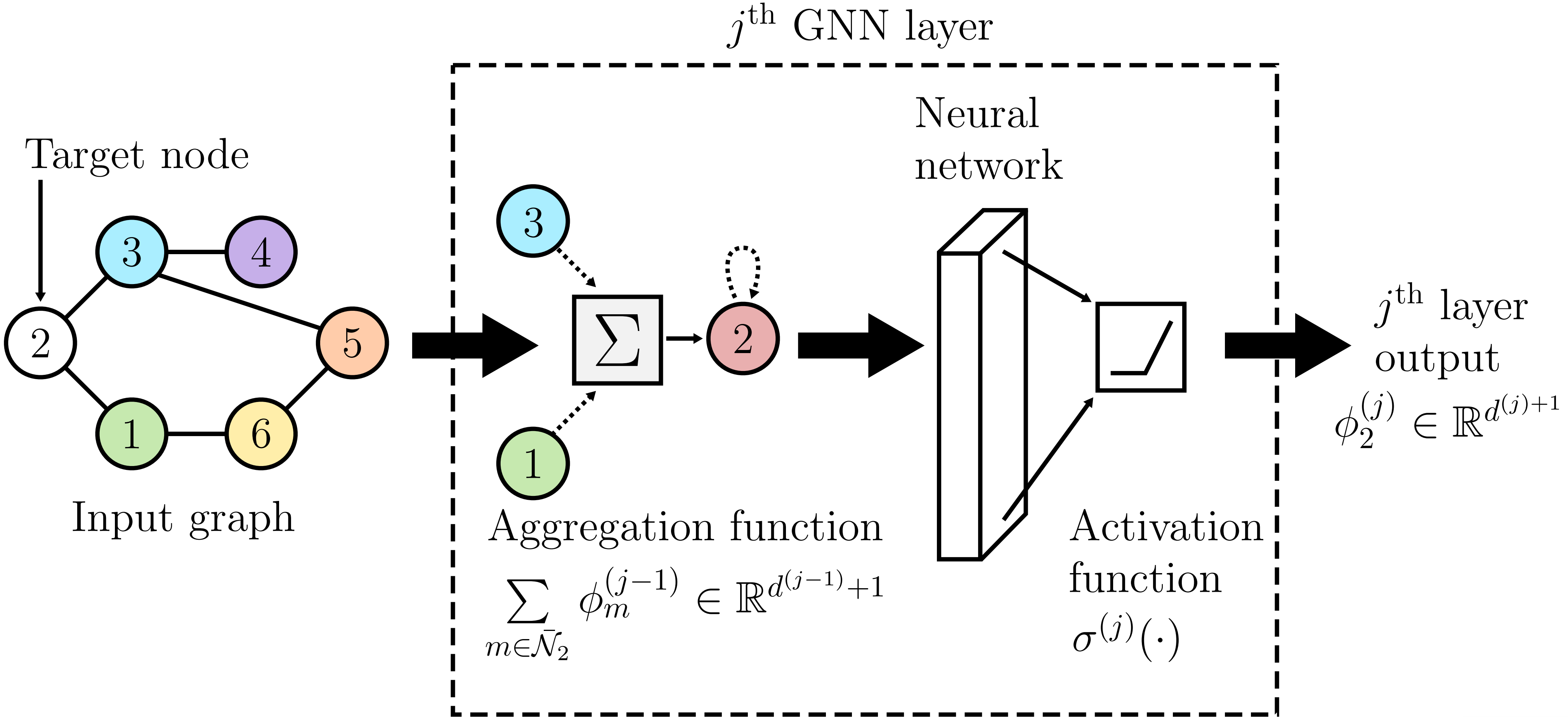


Credit: NCR Lab

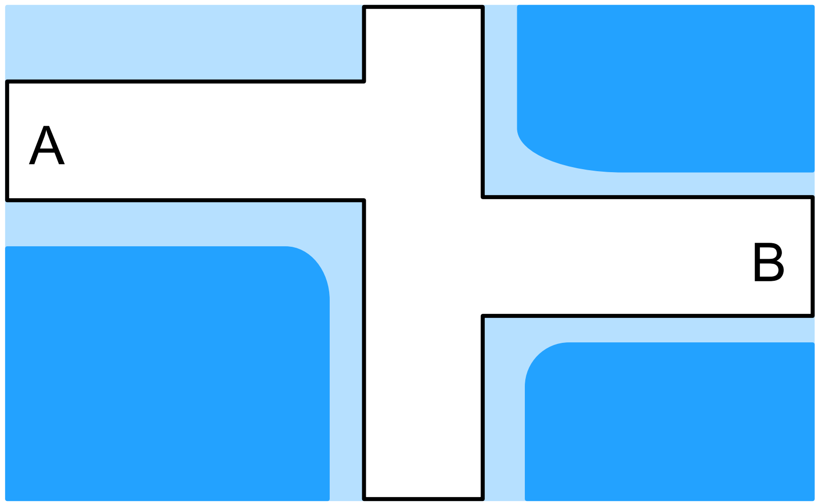


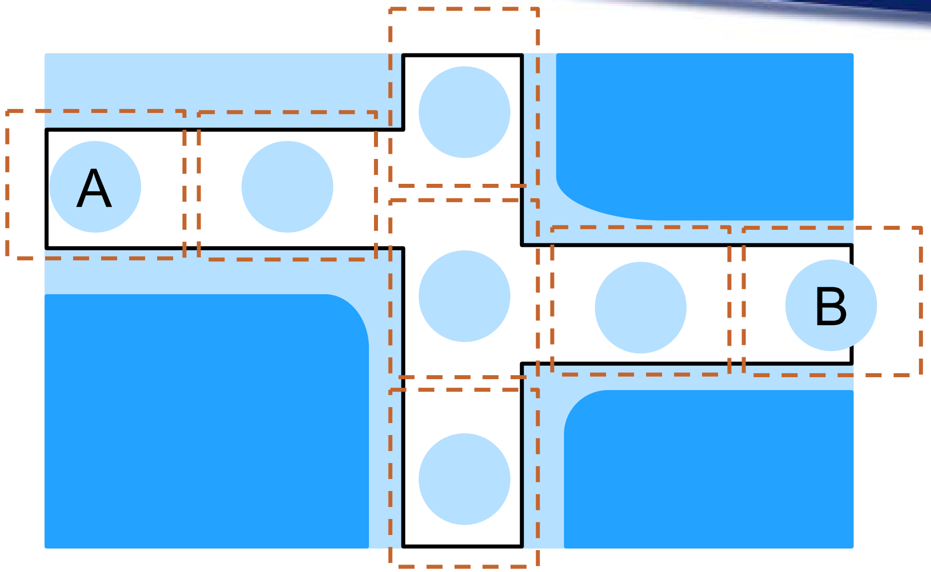
- MASs must communicate to accomplish cooperative goals
- Use estimates from neighboring agents to reach goal effectively


Target node
3
4
2
5
1
6
Input graph
2
Σ
3
Σ
5
2
4
1
Σ
2
6

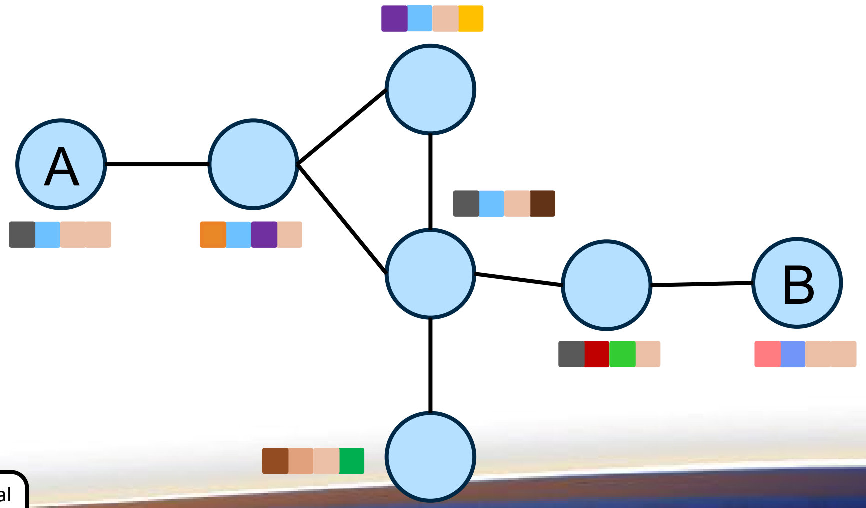

j^{th} GNN layer
Target node
3
4
2
5
1
6
Input graph
3
$\sum$
2
1
Neural network
Aggregation function
$\sum_{m \in \bar{\mathcal{N}}_2} \phi_m^{(j-1)} \in \mathbb{R}^{d^{(j-1)}+1}$
Activation function
$\sigma^{(j)}(\cdot)$
j^{th} layer output
$\phi_2^{(j)} \in \mathbb{R}^{d^{(j)}+1}$

GOOGLE MAPS USERS = GNN ENJOYERS

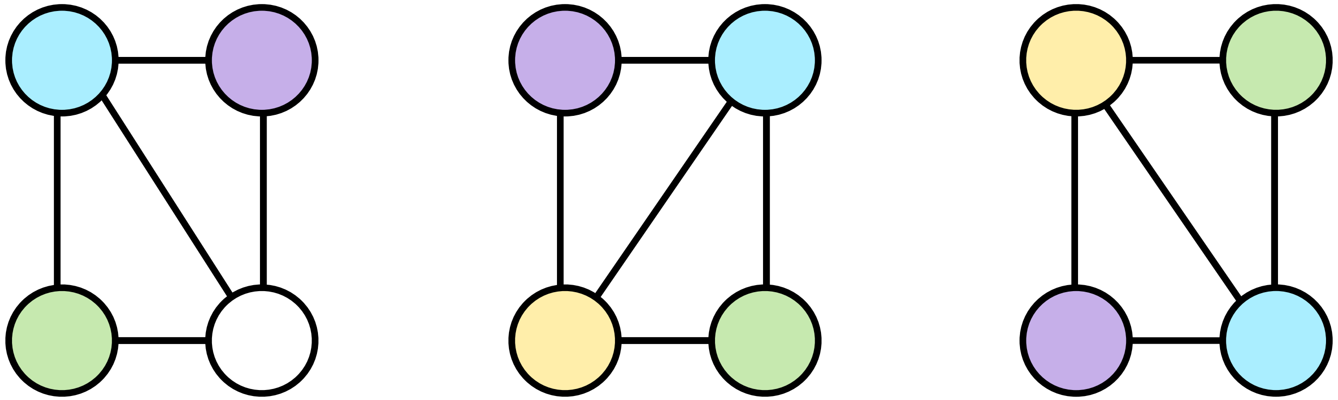




- Node-level info: anonymized historical segment travel speeds, segment length, and segment type (highway, state road, etc.)
- Train GNN to predict traversal time from A to B given the time of day [1]
- Global traversal time loss function



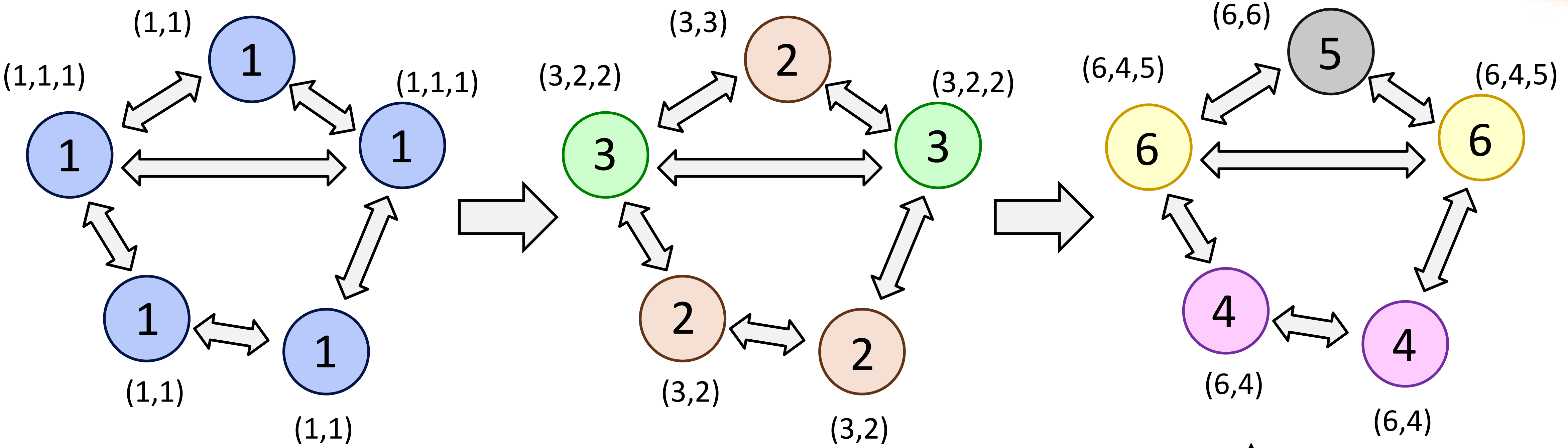
[1] Derrow-Pinion, A., et al, "Eta prediction with graph neural networks in google maps," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, 2021, pp. 3767–3776.



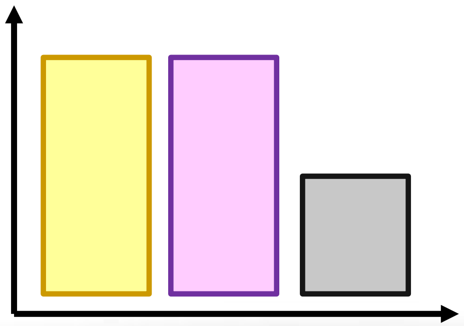
These graphs are **isomorphic**!

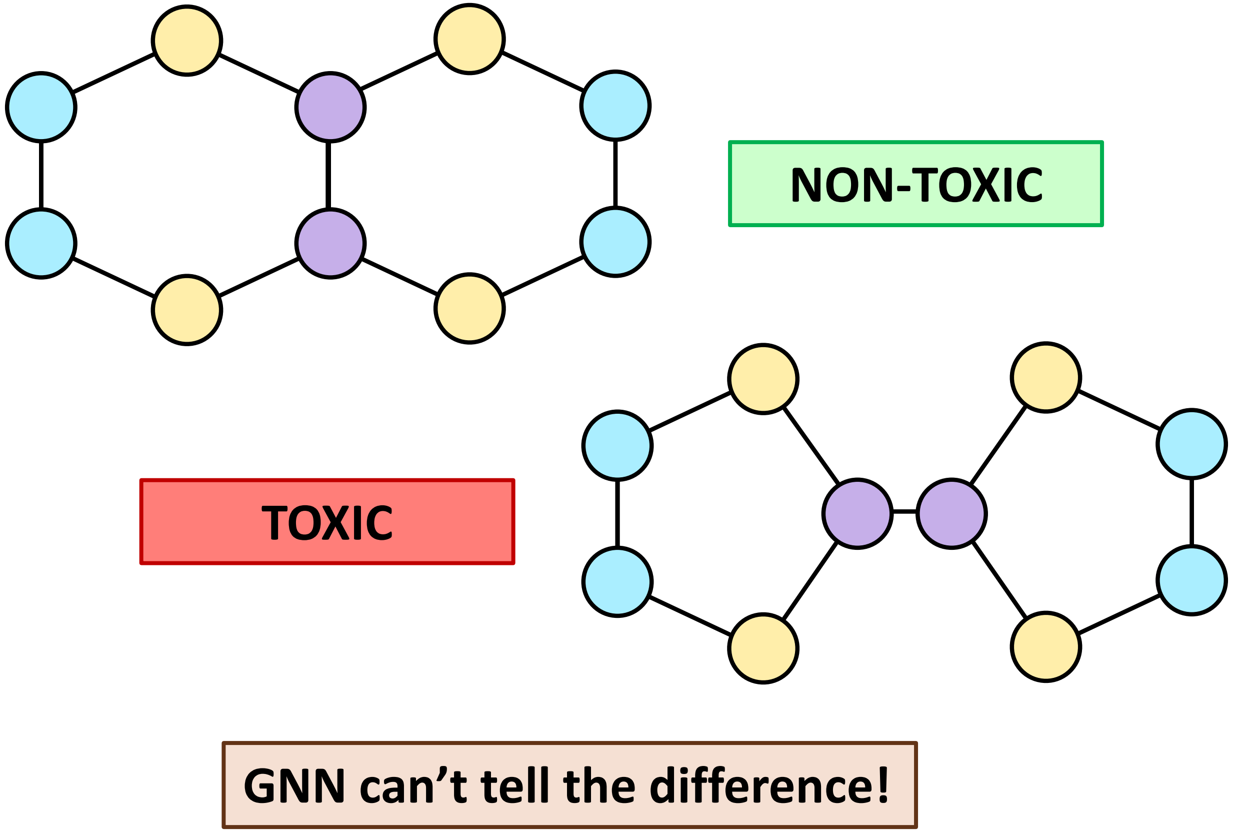
Open problem: Development of polynomial time algorithm to determine whether two graphs are isomorphic

- There exists a permutation that relates the nodes of graphs 1,2, and 3
- How can we test for this?



- 1-WL (Weisfeiler-Lehman) test
- Any two graphs that are isomorphic will have the same color distribution after 1-WL are isomorphic **(but not the other way around!)**
- Generalize to higher dimensions with k-WL (uses pairs, triples, …)

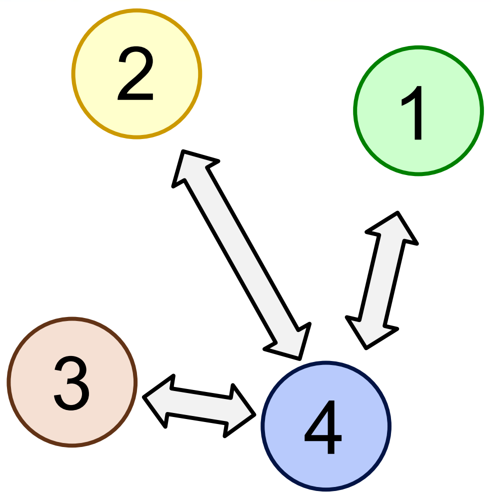




- Invariant function = function output the same, **regardless** of node order
- Equivariant function = function output **respects** node order
- Message-passing GNNs can distinguish up to 2-WL equivalence [2]

Continuous functions of each node's features on a graph are **equivariant**.

[2] W. Azizian and M. Lelarge, "Expressive power of invariant and equivariant graph neural networks," *Proc. Int. Conf. Learn. Represent.*, 2020.



- Graph attention network (GAT) architecture [3]
- Rank importance of messages

[3] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *Int. Conf. Learn. Represent.*, 2018.

Typically normalized using softmax

$$c_{i,\ell} = a_i^{(j)\top}\left(\left(W_i^{(j)\top}\phi_i^{(j-1)}\right) \oplus \left(W_i^{(j)\top}\phi_\ell^{(j-1)}\right)\right)$$

- Message passing structure introduced challenges in GNN derivative w.r.t. weights calculation
- Need to "chase down" your own weights in update law

$$\frac{\partial \phi_i}{\partial \mathrm{vec}\left(W_i^{(j)}\right)} = W_i^{(k)\top}\varphi_i^{(k-1)}$$

Recursively defined term in partial derivative w.r.t. weights

$$\varphi_{m^{(\ell+1)}}^{(\ell)} \triangleq \begin{cases} \Delta_{m^{(\ell+1)}}^{(\ell)}\left[\left(\varphi_{m^{(\ell)}}^{(\ell-1)}\right)^\top\right]_{\boldsymbol{m}^{(\ell)}}^\top, & \ell = k-1, \dots, j+1, \\ \delta_{i,m^{(\ell+1)}}\pi_{m^{(\ell+1)}}^{(\ell)}\iota_{m^{(\ell+1)}}^{(\ell)}, & \ell = j. \end{cases}$$

- Want network of agents to track target with unknown, unstructured dynamics

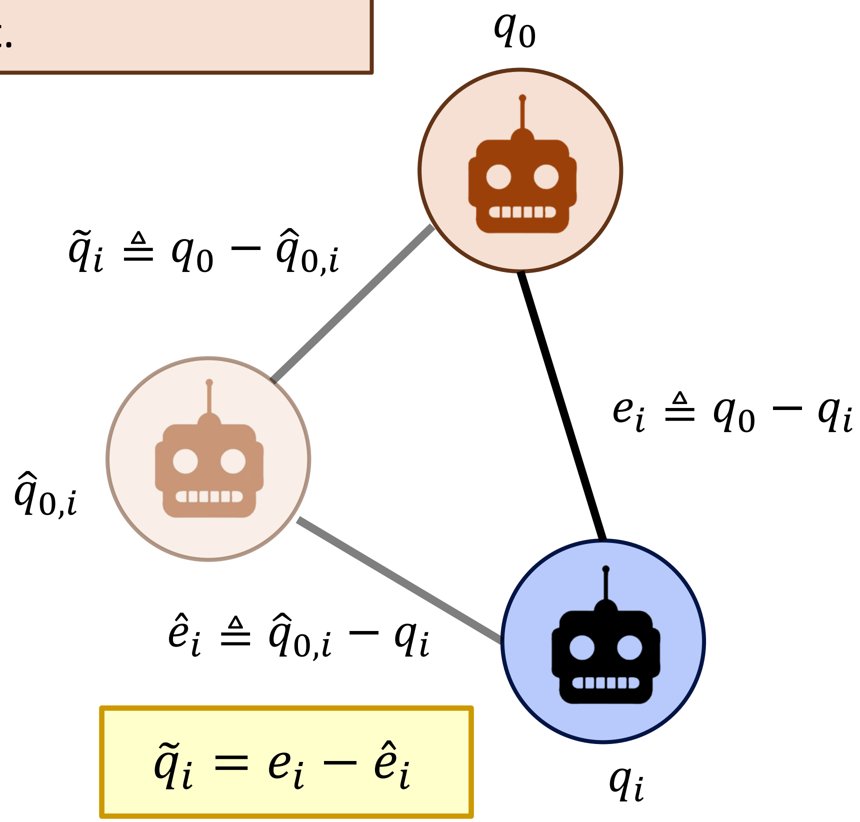
$$\ddot{q}_0 = f(Q_0),\ Q_0 = [q_0^\top, \quad \dot{q}_0^\top]^\top \in \mathbb{R}^{2n}$$

$$\ddot{q}_i = g(R_i) + u_i$$

Approximate together using GNNs

$$\boxed{r_{1,i} = \dot{\tilde{q}}_i + \alpha_1 \tilde{q}_i \qquad r_{2,i} = \dot{\hat{e}}_i + \alpha_2 \hat{e}_i}$$

Signals $e_i, \tilde{q}_\mathrm{i}$ are measurable iff $b_i = 1$. That is, if agent i is connected to the target.



$$\boxed{\tilde{q}_i = e_i - \hat{e}_i}$$

- Position tracking error (e_i) **"How far am I from the target?"**
- State estimation error $(\hat{e}_i)$ **"How far am I from my estimate?"**
- State estimation regulation error $(\tilde{q}_i)$ **"How far is my estimate from the target?"**

KEEPING ALL NODES ON THE SAME PAGE

$$\dot{\hat{\theta}}_{1,i} \triangleq \operatorname{proj}\left(\Gamma_i\left(-k_3\left(\sum_{j\in\mathcal{N}_i}(\hat{\theta}_{1,i}-\hat{\theta}_{1,j})+\hat{\theta}_i\right)+\sum_{j\in\bar{\mathcal{N}}_i^{k-1}}\frac{\partial\phi_{1,i}}{\partial\hat{\theta}_{1,j}}^{\top}\left(\sum_{j\in\mathcal{N}_i}(\dot{\hat{q}}_{0,j}-\dot{\hat{q}}_{0,i})+b_i\dot{\tilde{q}}_i+\alpha_1\sum_{j\in\mathcal{N}_i}(\hat{q}_{0,j}-\hat{q}_{0,i})+\alpha_1 b_i\tilde{q}_i\right)\right)\right)$$

Projection Operator:
Ensures that updated NN weights are bounded wrt $\bar{\theta}$.

Distributed Adaptation:
Performs "consensus in the weights" between nodes of the GNN.

σ-Modification Term:
Grants parameter convergence to within a neighborhood of ideal values.

NN Derivative w.r.t. Weights:
Closed-form derivative of GNN or GAT architecture w.r.t. weights.

Loss function:
Multiplies NN partial derivative by an implementable form of the auxiliary state estimation regulation error ("How far is my estimate from the target?"). This is a term we wish to minimize.

"Distributed Adaptation Law"

- Nodes cannot perform backpropagation at the same time with the same set of info
- Every node has its own unique set of weights
- We want them to converge to the same values (we are all approximating the same unknown function!)

$\hat{Q}_{0,i} \triangleq [\hat{q}_{0,i}^{\top}, \dot{\hat{q}}_{0,i}^{\top}]^{\top} \in \mathbb{R}^{2n}$

$$\ddot{\hat{q}}_{0,i} \triangleq \phi_{1,i}(\hat{Q}_{0,i}) + \sum_{j\in\bar{\mathcal{N}}_i^k} \frac{\partial \phi_{1,i}}{\partial \hat{\theta}_{1,j}}^{\top} + k_1\left(\sum_{j\in\mathcal{N}_i}(\dot{\hat{q}}_{0,j} - \dot{\hat{q}}_{0,i}) + b_i\dot{\tilde{q}}_i + \alpha_1\sum_{j\in\mathcal{N}_i}(\hat{q}_{0,j} - \hat{q}_{0,i}) + \alpha_1 b_i \tilde{q}_i\right)$$

GNN Output:
Used to approximate unknown target dynamics at each node.

Distributed Adaptation:
Cancel Taylor expansion terms due to influence of neighboring weights.

Observer Update Law

Error signal:
Implementable form of the auxiliary state estimation regulation error. ("How far is my estimate from the target?")

$$u_i \triangleq \ddot{\hat{q}}_{0,i} - \phi_{2,i}(R_i) - \sum_{j\in\bar{\mathcal{N}}_i^k} \frac{\partial \phi_{2,i}}{\partial \hat{\theta}_{2,j}}^{\top} + k_2\left(\sum_{j\in\mathcal{N}_i}(\dot{\hat{e}}_i - \dot{\hat{e}}_j) + b_i\dot{\hat{e}}_i + \alpha_1\sum_{j\in\mathcal{N}_i}(\hat{e}_i - \hat{e}_j) + \alpha_1 b_i \hat{e}_i\right)$$

- Observer = drive estimate to the target
- Controller = drive agent to estimate

Observer:
Accounts for GNN estimate of target motion to inform control update.

Error signal:
Implementable form of the auxiliary state estimation error. ("How far am I from my estimate?")

Control Law

$R_i \triangleq \left[\mathbf{1}_{\bar{\mathcal{N}}_i}(m) Q_m^{\top}\right]_{m\in V}^{\top} \in \mathbb{R}^{2n}N$

- Consider a candidate Lyapunov function

$$V = \frac{1}{2}\tilde{q}^\top\tilde{q} + \frac{1}{2}\hat{e}^\top\hat{e} + \frac{1}{2}r_1^\top\mathcal{H}r_1 + \frac{1}{2}r_2^\top r_2 + \frac{1}{2}\tilde{\theta}_1^\top\Gamma_1^{-1}\tilde{\theta}_1 + \frac{1}{2}\tilde{\theta}_2^\top\Gamma_2^{-1}\tilde{\theta}_2$$

Theorem 1 (Stability Result)

For agent and target dynamics described on Slide 18 and initial conditions of the states $\zeta(t_0) \in \mathcal{S}$, the observer, controller, and adaptive update law guarantee that ζ exponentially converges to $\mathcal{U}$ where

$$\|\zeta(t)\| \leq \left(\frac{\lambda_2}{\lambda_1}\left(\frac{\upsilon}{\lambda_4} + e^{-\frac{\lambda_4}{\lambda_2}(t-t_0)}\left(\|\zeta(t_0)\|^2 - \frac{\upsilon}{\lambda_4}\right)\right)\right)^{\frac{1}{2}},$$

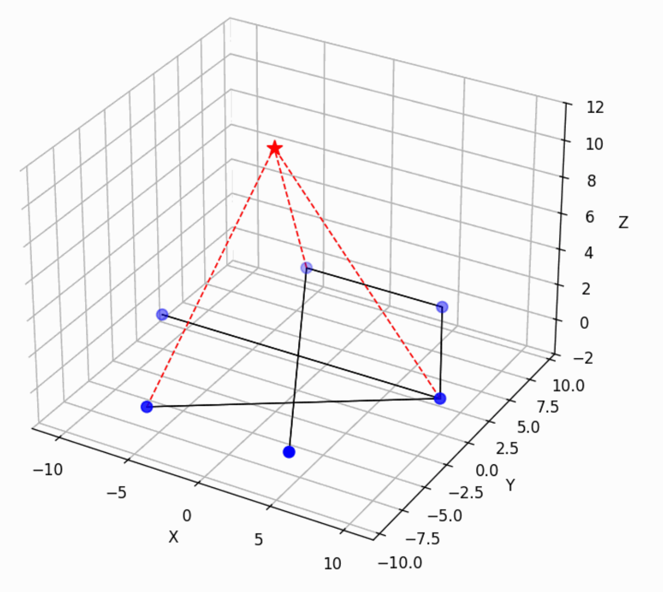
for all $t \in \mathbb{R}_{\geq 0}$ given that the constants and control gains $\alpha_1, \alpha_2, \epsilon_1, \epsilon_2, k_1, k_2, k_3$, and λ_3 are chosen according to their respective sufficient conditions.

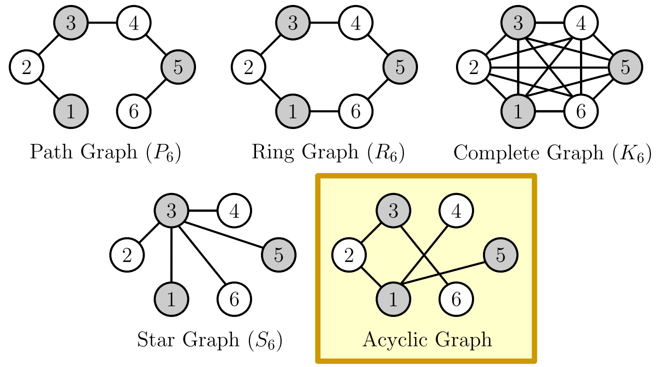
- $N = 6$ agents, 3 agents connected to target agent
- Unknown target dynamics of the form

$$\begin{bmatrix}\ddot{x}_0\\ \ddot{y}_0\\ \ddot{z}_0\end{bmatrix} = \begin{bmatrix}\cos(\dot{x}_0) - \sin(\dot{y}_0) + \cos(2\dot{z}_0)\\ \dot{x}_0 - \dot{y}_0 + \dot{z}_0 + \frac{y_0}{\sqrt{1+|y_0|}}\\ \sin(\dot{y}_0) - \dot{x}_0\dot{z}_0\end{bmatrix}$$

- Unknown inter-agent dynamics of the form

$$\begin{bmatrix}\ddot{x}_i\\ \ddot{y}_i\\ \ddot{z}_i\end{bmatrix} = \begin{bmatrix}\sum_{j\in\mathcal{N}_i}\frac{1}{20{,}000(y_i-y_j)^2}\\ \sum_{j\in\mathcal{N}_i}(\dot{z}_i-\dot{z}_j)\cos(\dot{x}_i)\\ \sum_{j\in\mathcal{N}_i}\frac{\cos(\dot{z}_i\dot{z}_j)(\dot{x}_i-\dot{x}_j)}{\sqrt{1+|\dot{x}_i-\dot{x}_j|}}\end{bmatrix} + u_i$$





Architecture	e_{RMS}	$\dot{e}_{RMS}$	$\tilde{\Phi}_{1,RMS}[0{:}10]$	$\tilde{\Phi}_{1,RMS}[10{:}60]$	$\tilde{\Phi}_{2,RMS}[0{:}10]$	$\tilde{\Phi}_{2,RMS}[10{:}60]$	u_{RMS}
DNN+DNN	0.4844	0.4355	1.049	0.7635	2.430	0.1296	1.305
GNN+GNN	0.3952	0.4250	1.138	0.7580	2.169	0.0805	1.405
GAT+GNN	0.2912	0.3899	2.676	0.5684	2.246	0.0649	1.570

49% improvement in tracking error performance over DNN baseline!

Thank you! Any questions?