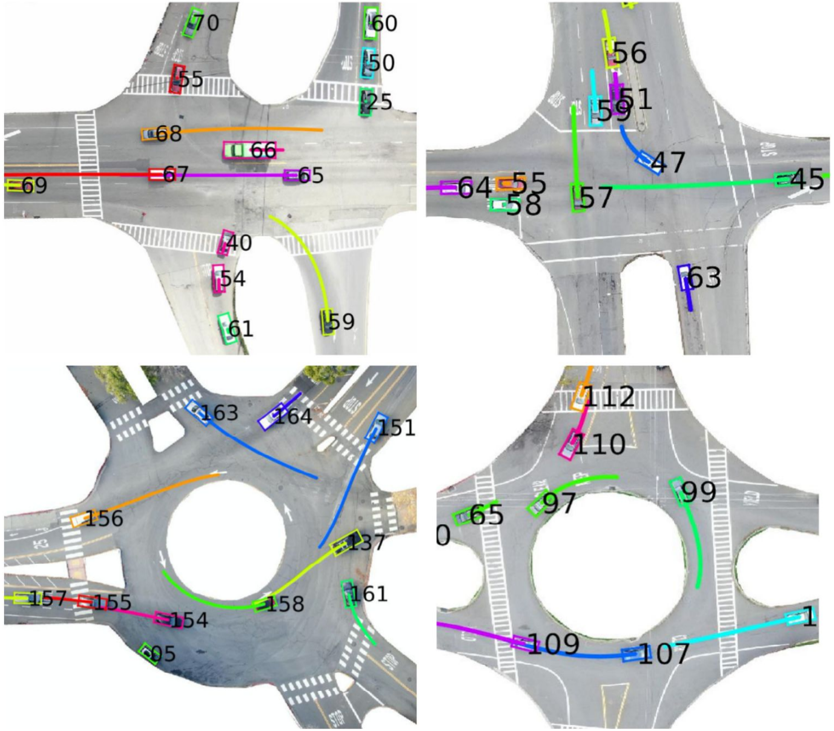
Neural Relational Inference with Fast Modular Meta-learning

Improvements to modular techniques for modeling interacting systems with little data

Ferran Alet, Erica Weng, Tomas Lozano-Perez, Leslie Pack Kaelbling

Modeling Interacting Systems



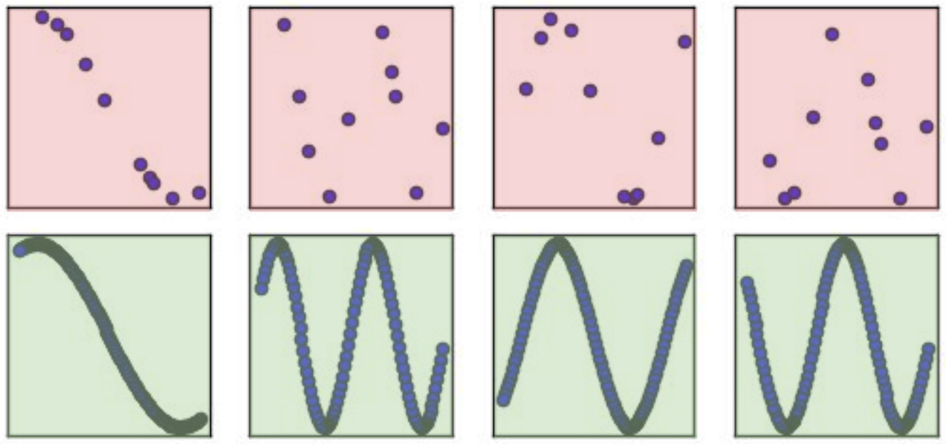
Background: Modular Meta-learning

Meta-learning

learns characteristics shared by similar tasks

Train

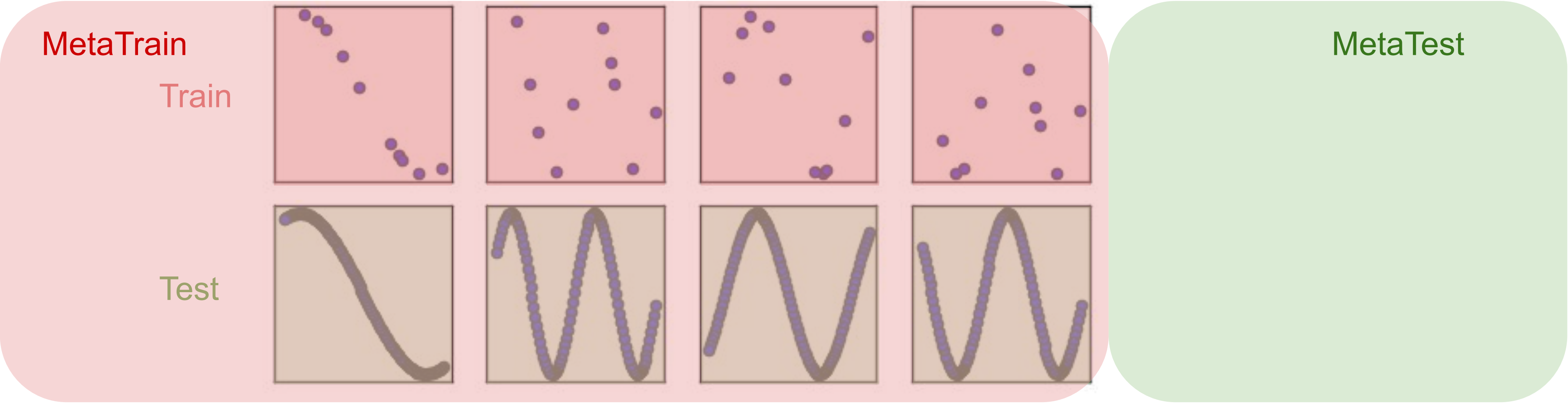
Test



Adapted from Finn et al.

Meta-learning

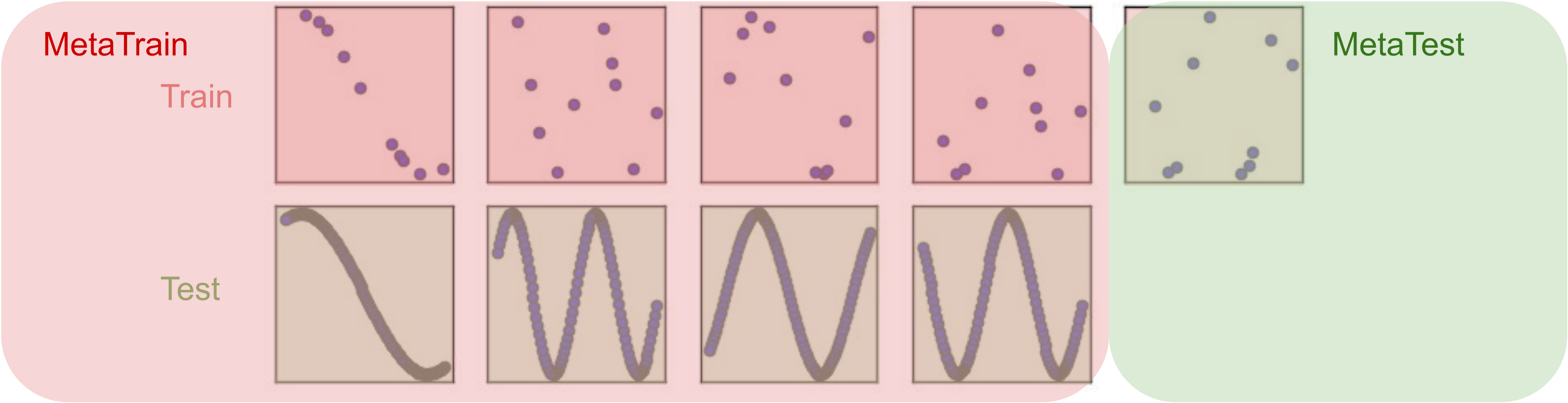
learns characteristics shared by similar tasks



Adapted from Finn et al.

Meta-learning

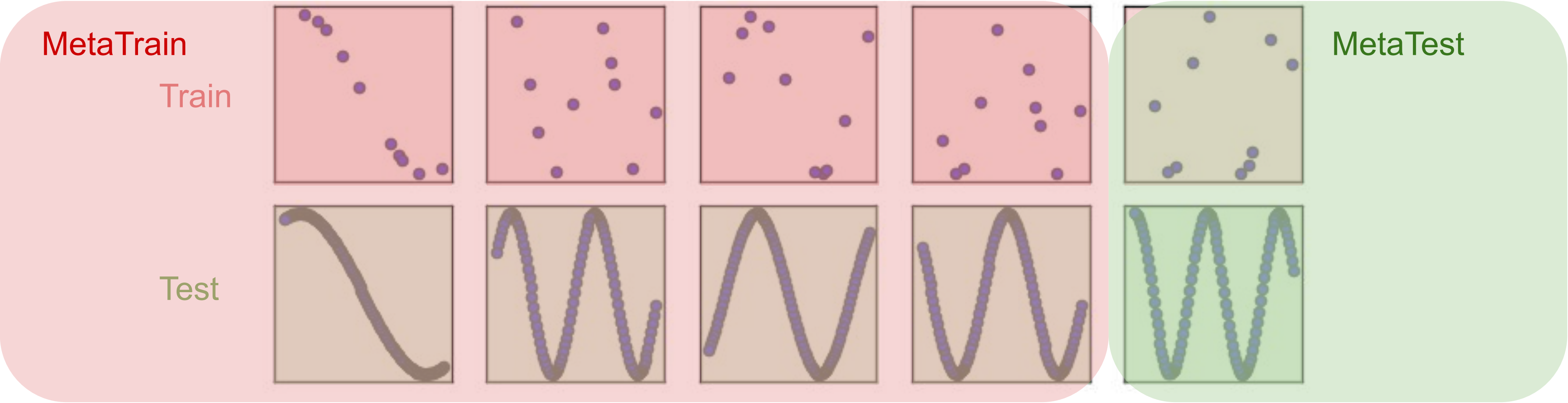
learns characteristics shared by similar tasks



Adapted from Finn et al.

Meta-learning

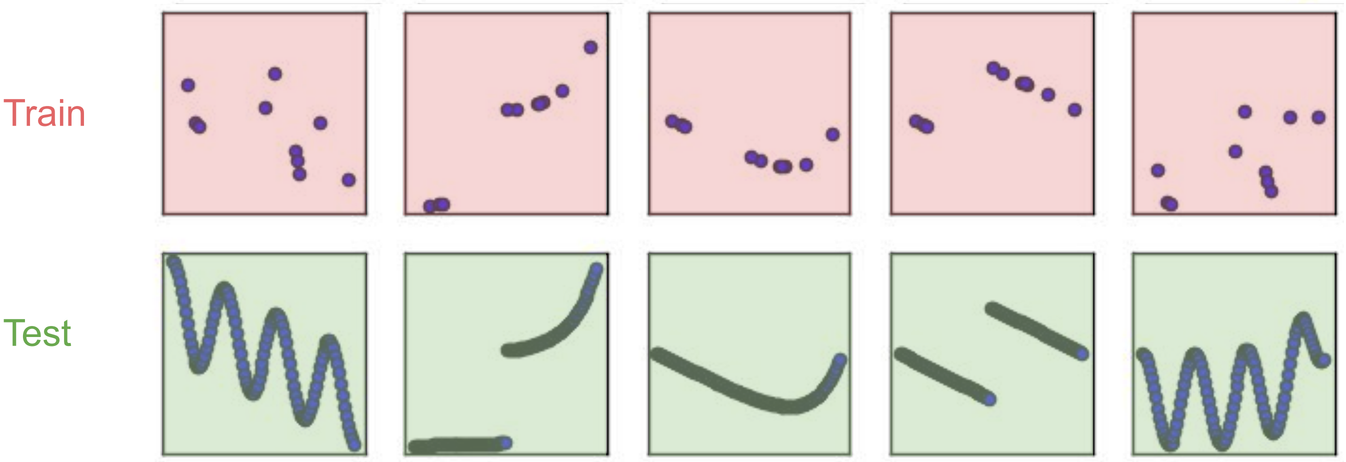
learns characteristics shared by similar tasks



Adapted from Finn et al.

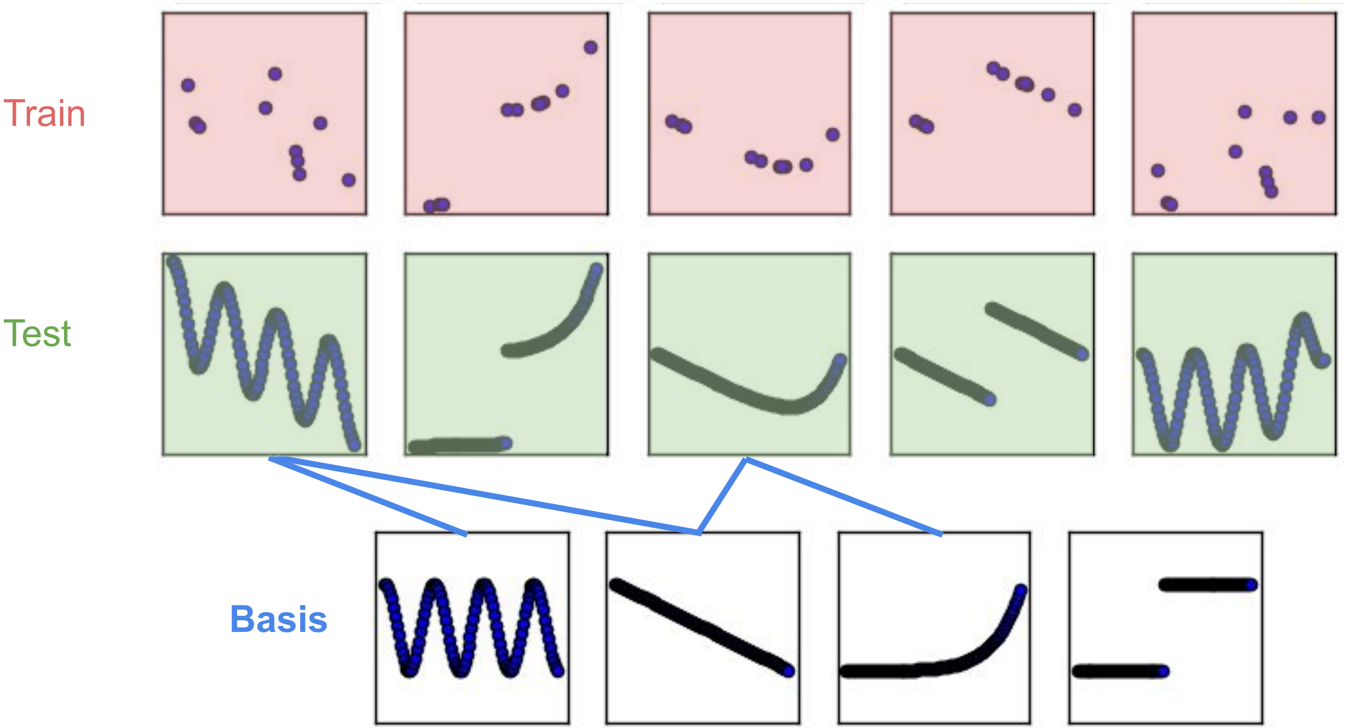
Modular meta-learning

learns a *modular decomposition* of characteristics shared by similar tasks



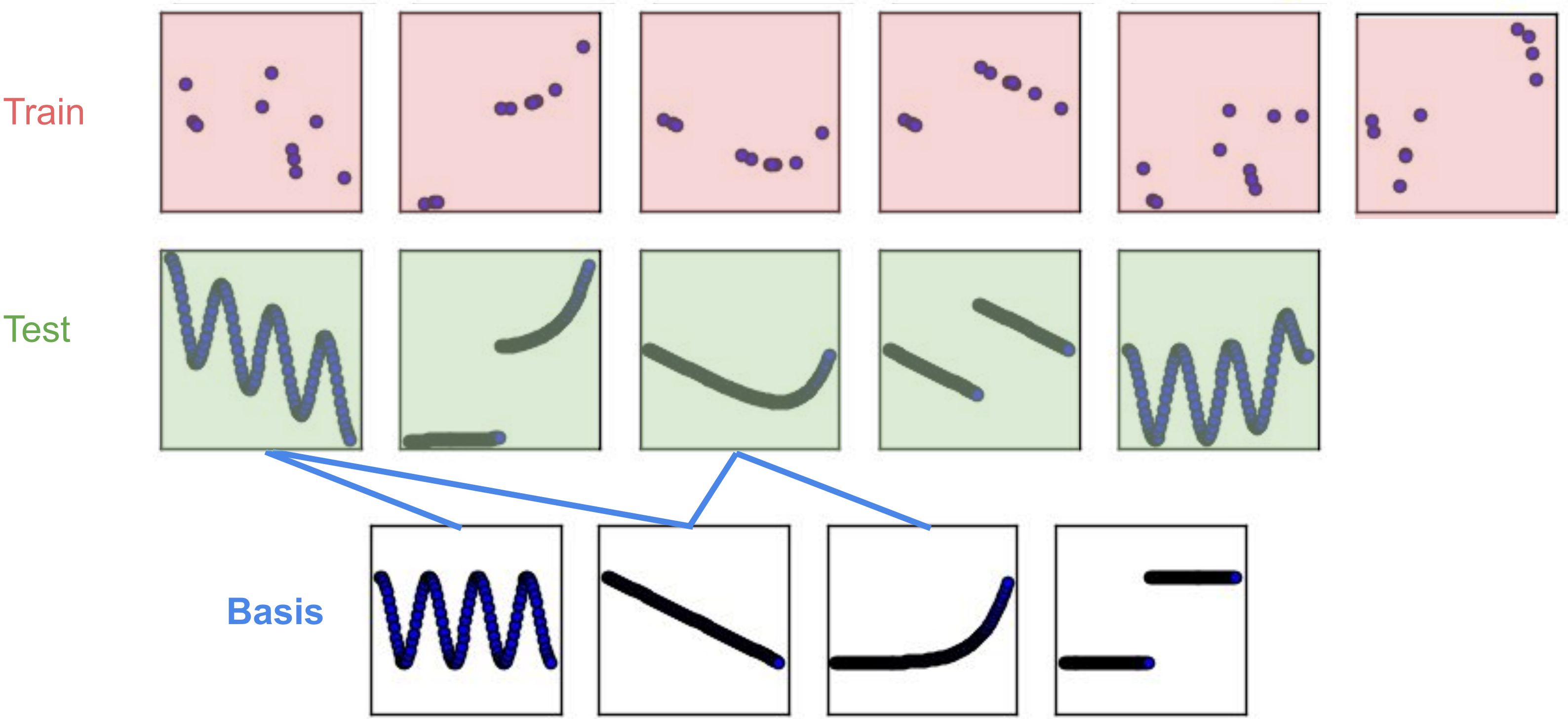
Modular meta-learning

learns a *modular decomposition* of characteristics shared by similar tasks



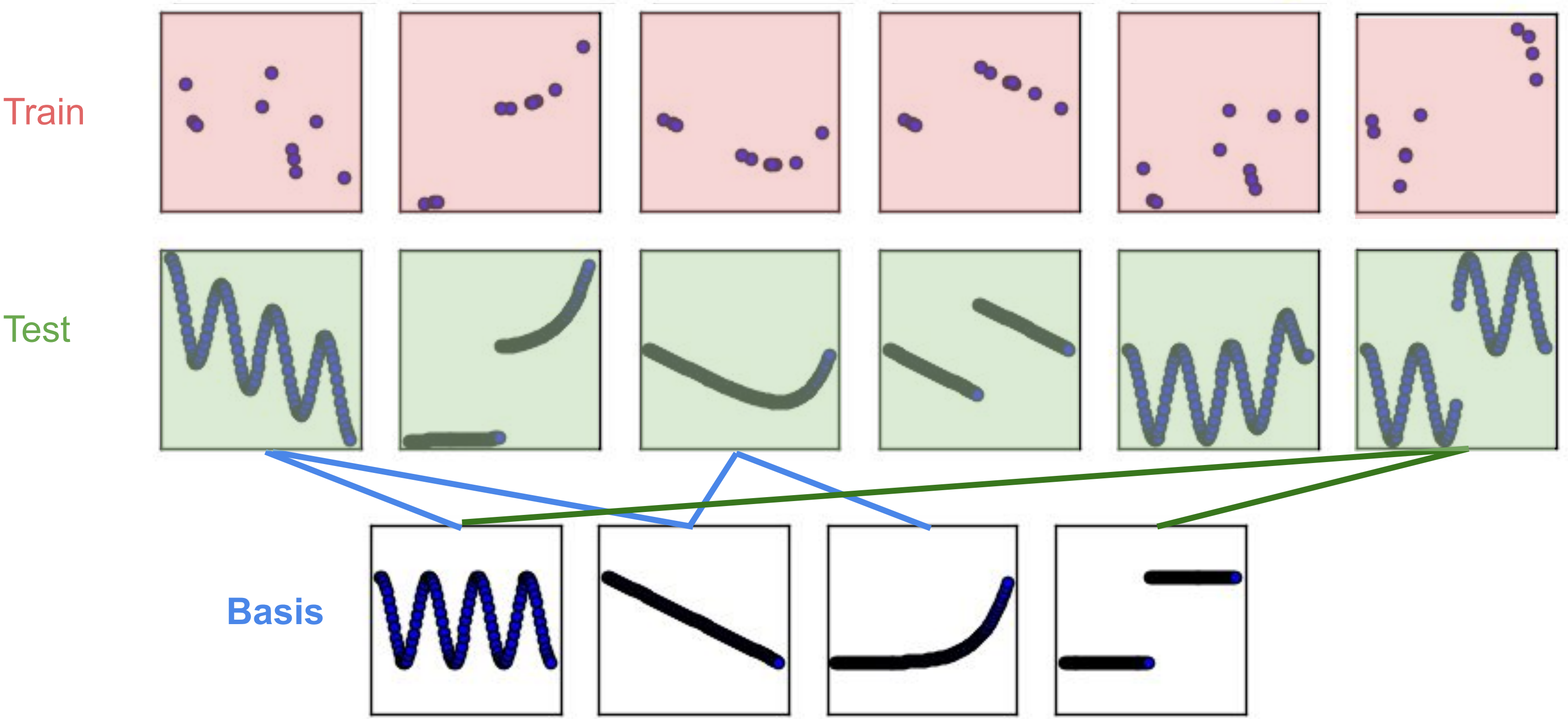
Modular meta-learning

learns a *modular decomposition* of characteristics shared by similar tasks



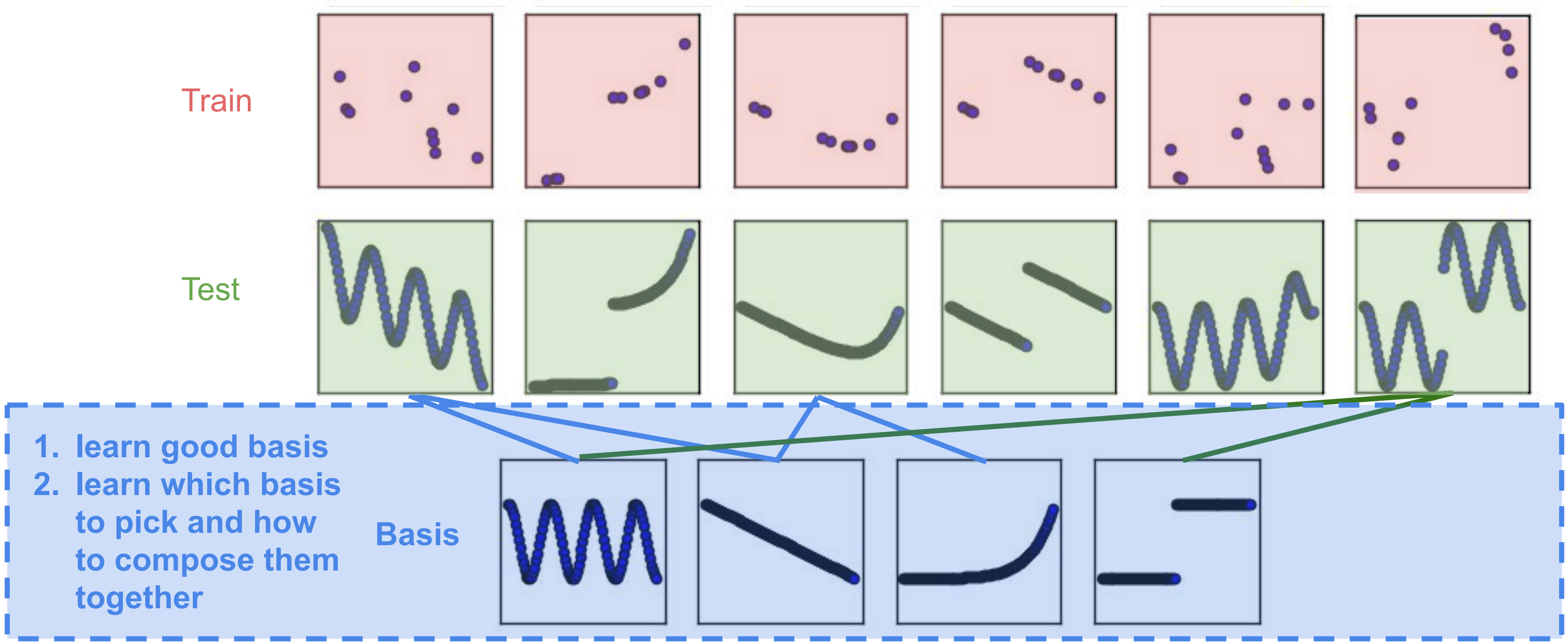
Modular meta-learning

learns a *modular decomposition* of characteristics shared by similar tasks



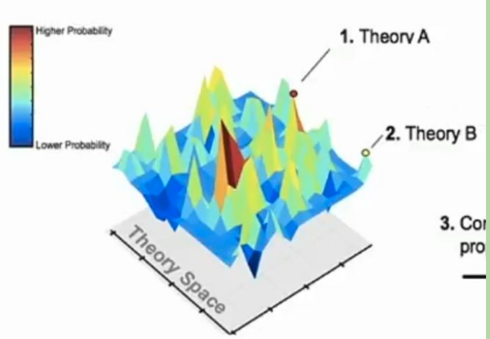
Modular meta-learning

learns a *modular decomposition* of characteristics shared by similar tasks

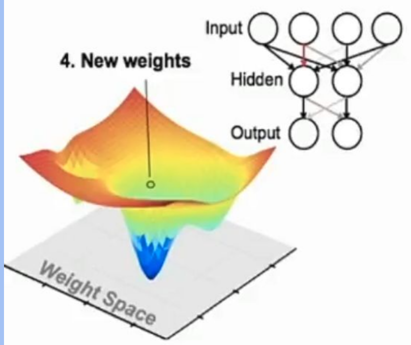


1. How to learn good modules?

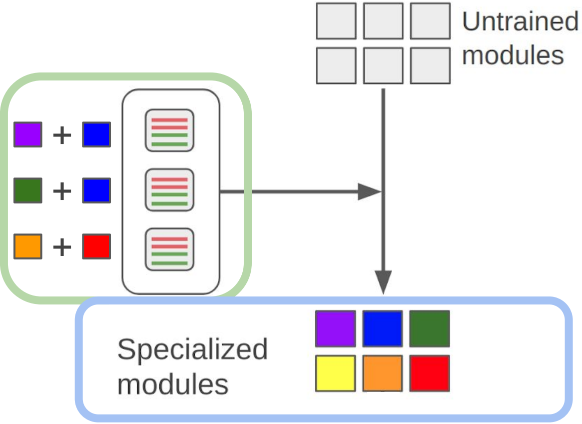
Best structure for each dataset



Good module weights for all datasets



Modular meta-learning



Source images: Josh Tenenbaum

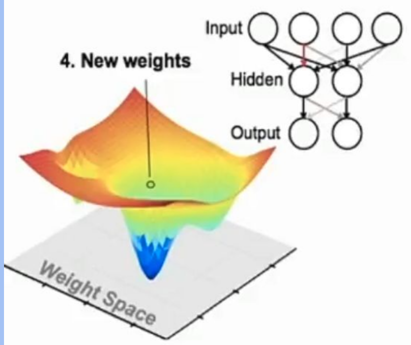
BounceGrad

Best structure
for each dataset

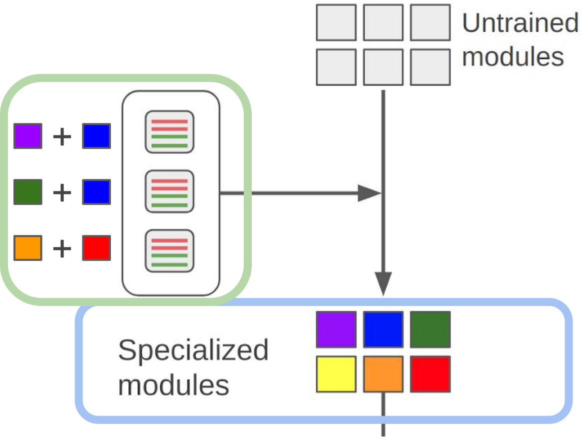
procedure BOUNCE($S_1, \ldots, S_m, D_1^{train}, \ldots, D_m^{train}, T, \mathcal{S}, \Theta$)
  for $j = 1 \ldots m$ **do**
    $S'_j = Propose_{\mathcal{S}}(S_j, \Theta)$
    if $Accept(e(D_j^{train}, S'_j, \Theta), e(D_j^{train}, S_j, \Theta), T)$ **then** $S_j = S'_j$

Simulated Annealing

Good module weights
for all datasets



Modular meta-learning



2a. How to compose them together?

BounceGrad

Best structure for each dataset

procedure BOUNCE($S_1, \dots, S_m, D_1^{train}, \dots, D_m^{train}, T, \mathcal{S}, \Theta$)
  for $j = 1 \dots m$ **do**
    $S'_j = Propose_{\mathcal{S}}(S_j, \Theta)$
    if $Accept(e(D_j^{train}, S'_j, \Theta), e(D_j^{train}, S_j, \Theta), T)$ **then** $S_j = S'_j$

Slow

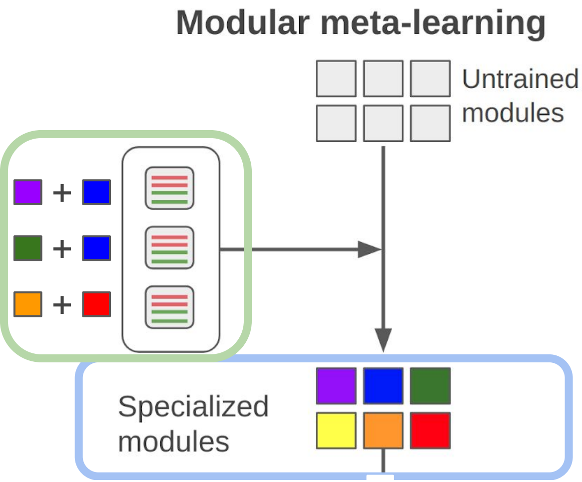
Simulated Annealing

Good module weights for all datasets

procedure GRAD($\Theta, S_1, \dots, S_m, D_1^{val}, \dots, D_m^{val}, \eta$)
  $\Delta = 0$
  for $j = 1 \dots m$ **do**
    $(x, y) = \text{rand\_elt}(D_j^{val})$; $\Delta = \Delta + \nabla_\Theta L(S_{j_\Theta}(x), y)$
  $\Theta = \Theta - \eta\Delta$

Fast

Gradient Descent



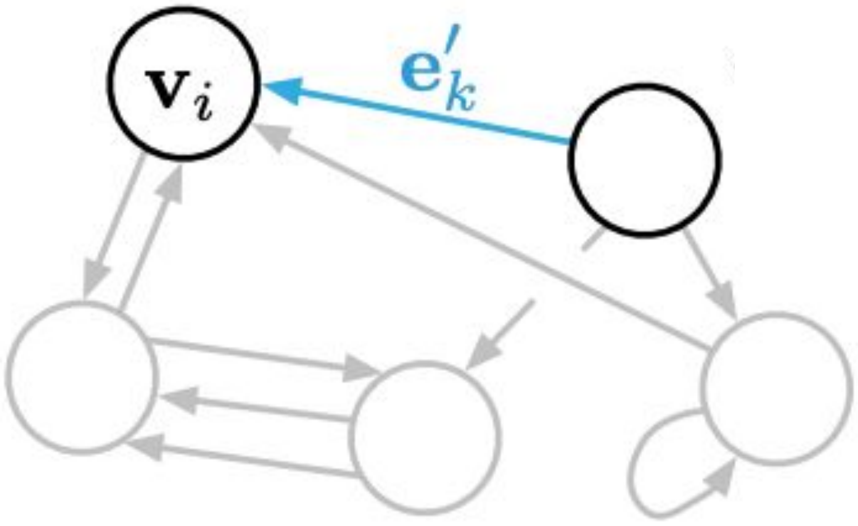
2b. How to compose them together?

Given modules f(x), g(x), h(x) there are many ways to compose them

- Sum: f(x) + g(x)
- Composition f(g(h(x)))
- Concatenation [f(x), g(x), h(x)]
- **Nodes and edges in a Graph Neural Network**

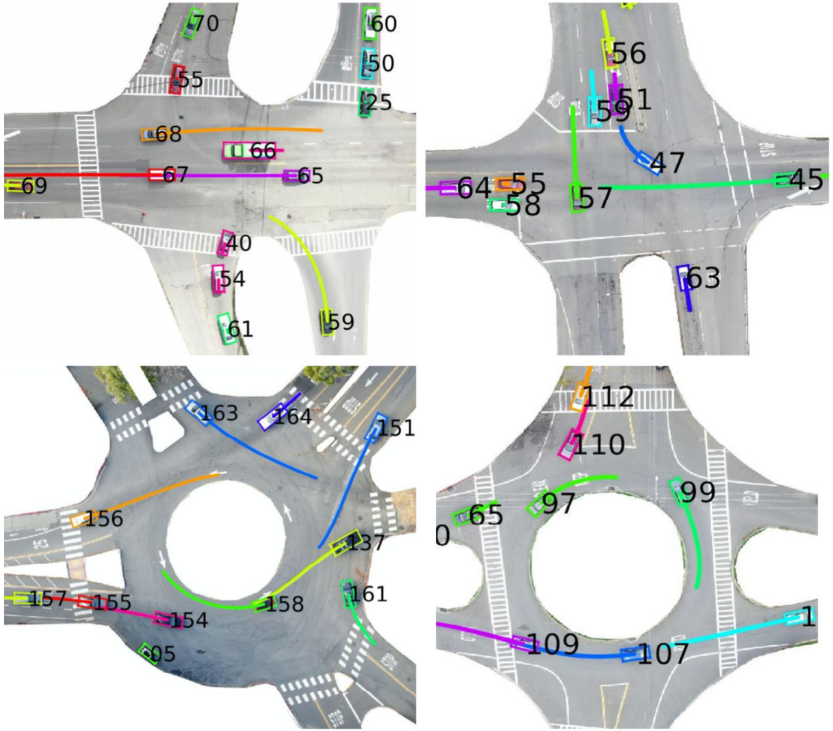
Background: Graph Neural Networks

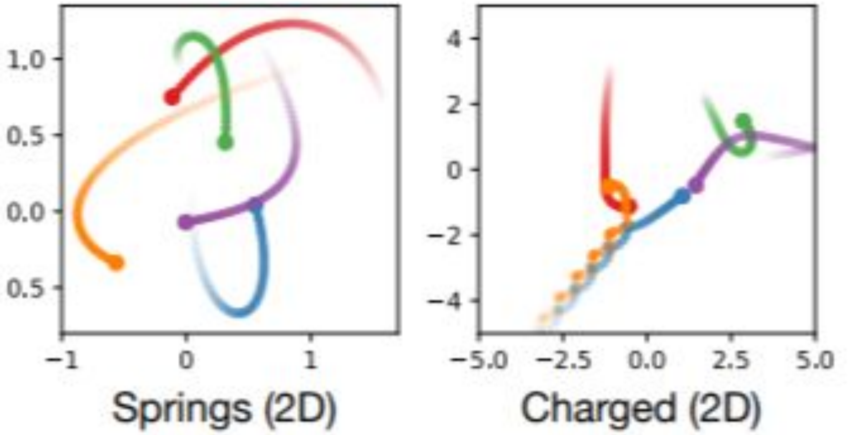
Graph Neural Networks



1. node and edge modules reused across the graph
2. similar inductive bias to CNNs

Modeling Interacting Systems





Modeling Physical Systems

$$x_1(t=0), x_1(t=1), \ldots, x_1(t=T)$$

$$x_2(t=0), x_2(t=1), \ldots, x_2(t=T)$$

$$\ldots$$

$$x_n(t=0), x_n(t=1), \ldots, x_n(t=T)$$

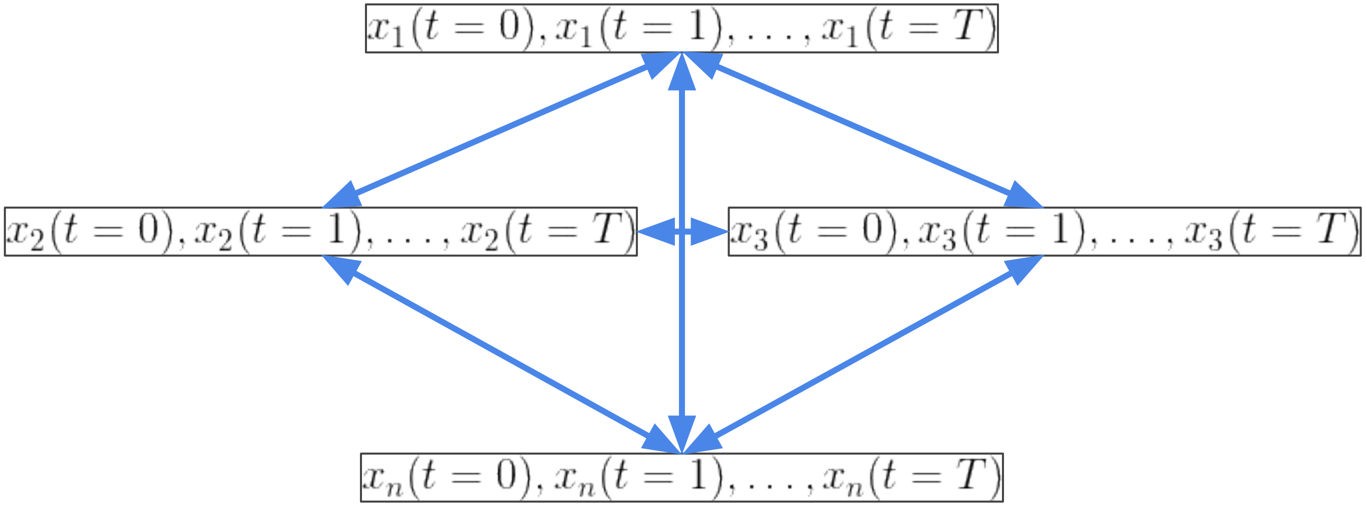
$$x_1(t=T+1), x_1(t=T+2), \ldots, x_1(t=T+k)$$

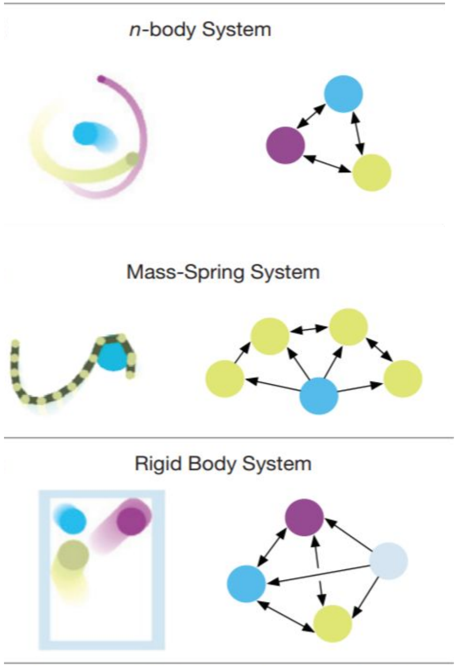
$$x_2(t=T+1), x_2(t=T+2), \ldots, x_2(t=T+k)$$

$$\ldots$$

$$x_n(t=T+1), x_n(t=T+2), \ldots, x_n(t=T+k)$$

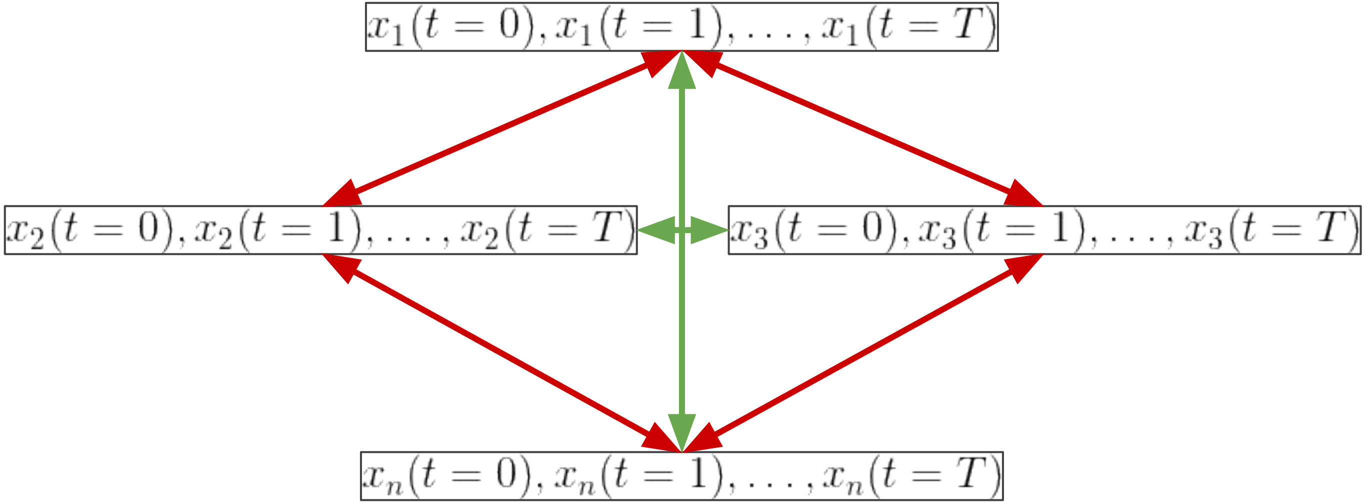
Modeling Physical Systems with Graphs

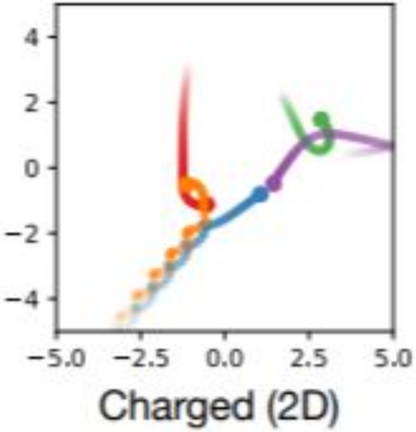




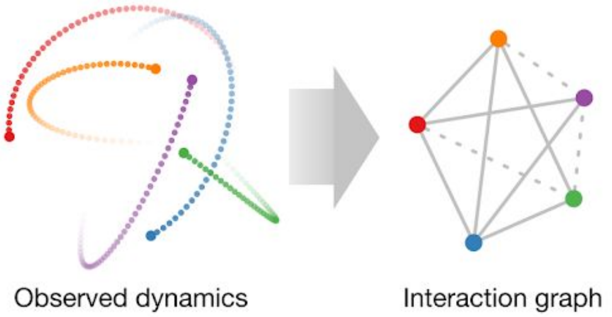
Modeling Physical Systems with Graphs

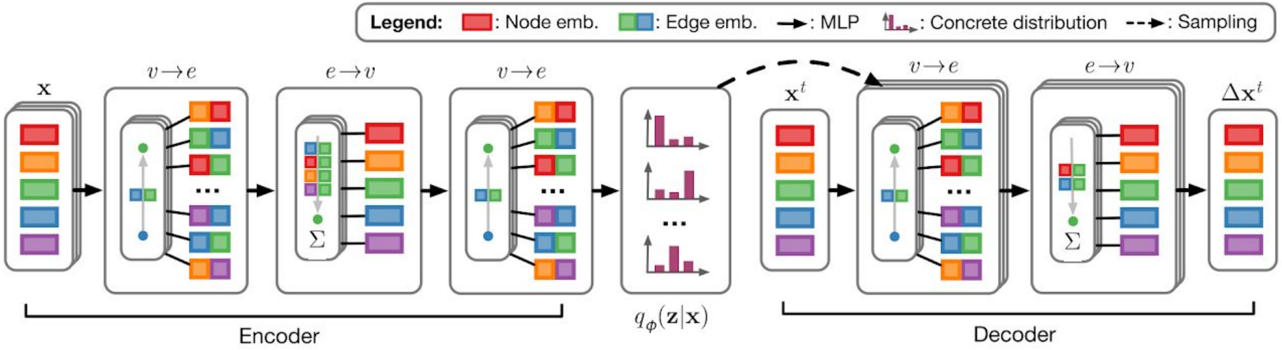
multiple interaction types





Neural Relational Inference (Kipf et al.)

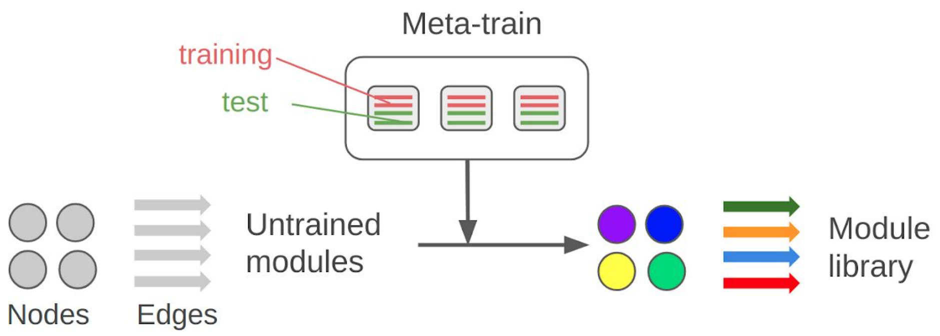




Fully connected GNN with 1 edge type

GNN where each directed edge is one of k types

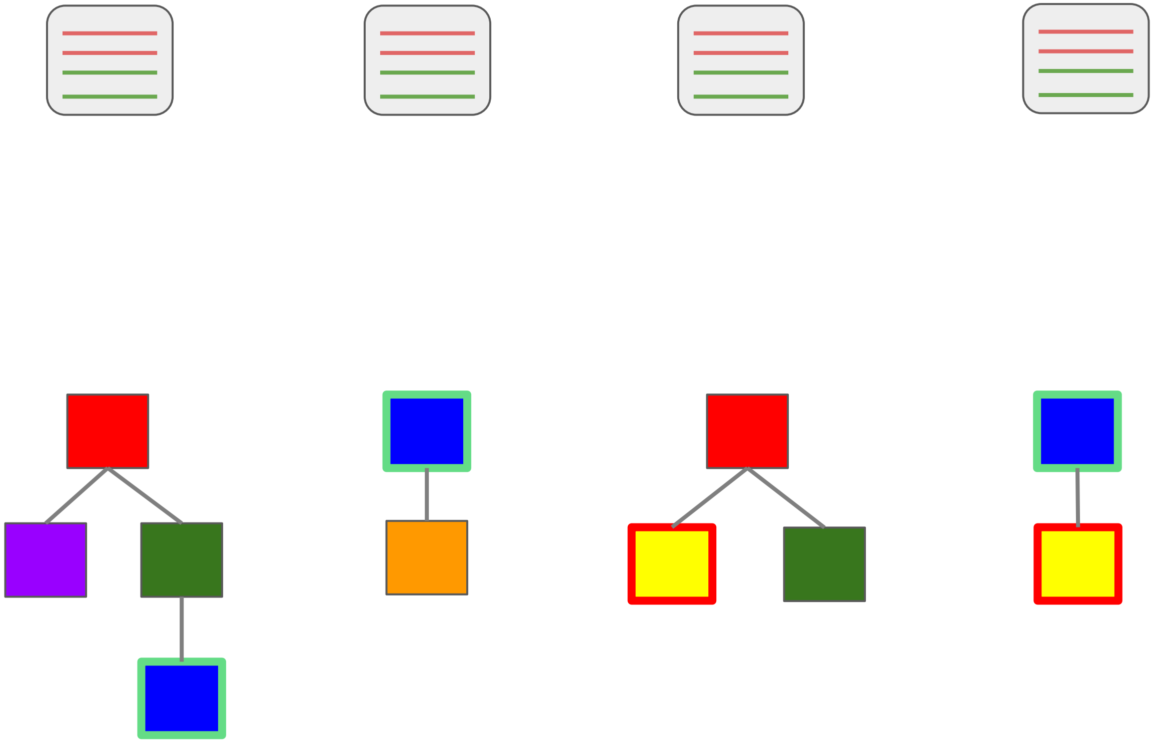
Neural Relational Inference as Modular Meta-learning



Original modular meta-learning is very slow

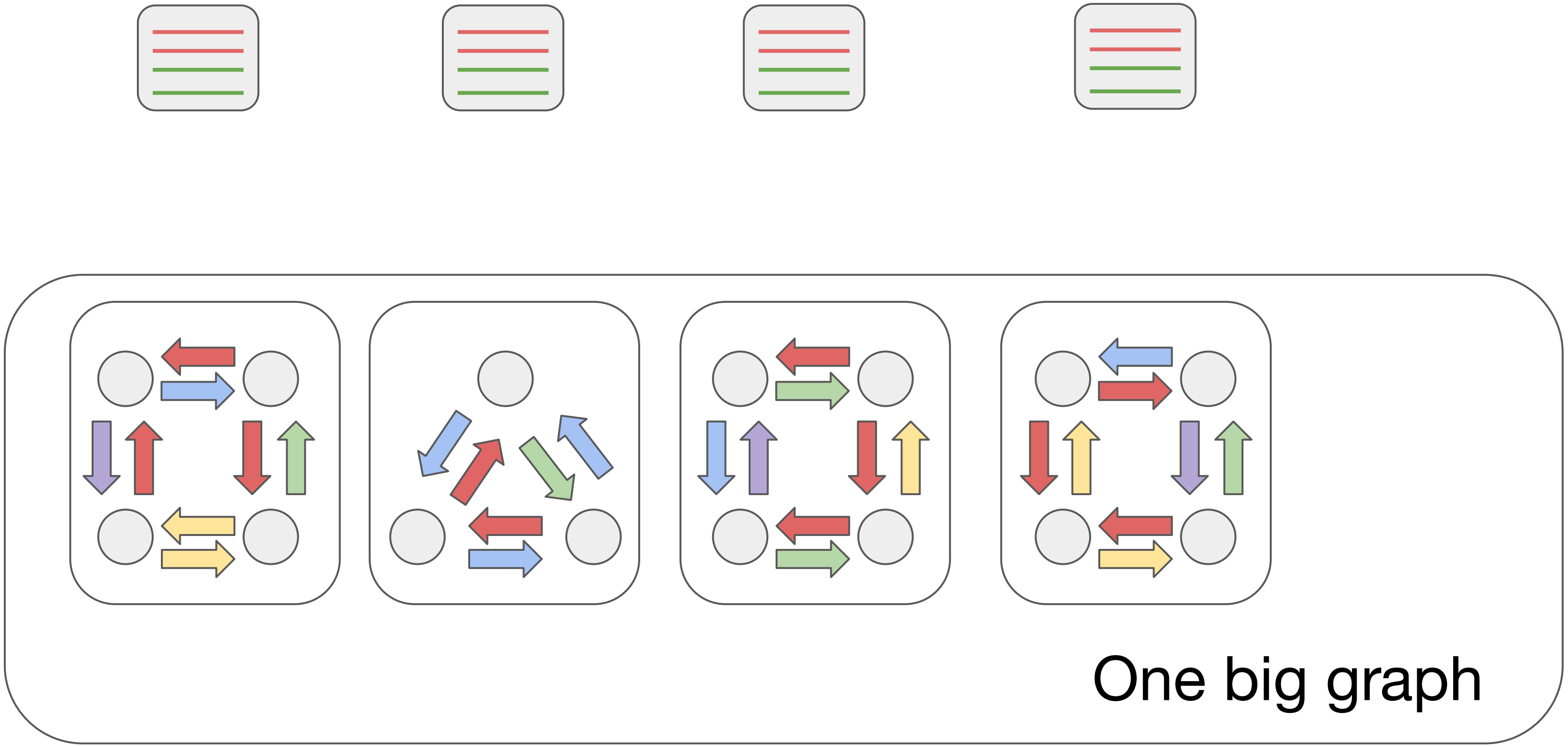
- Simulated Annealing makes bad proposals most of the time
- 200 datasets (CoRL 2018) → 50,000 datasets (NeurIPS 2019)
- Makes modular meta-learning a feasible approach for real applications (e.g. cars)

Batching multiple datasets



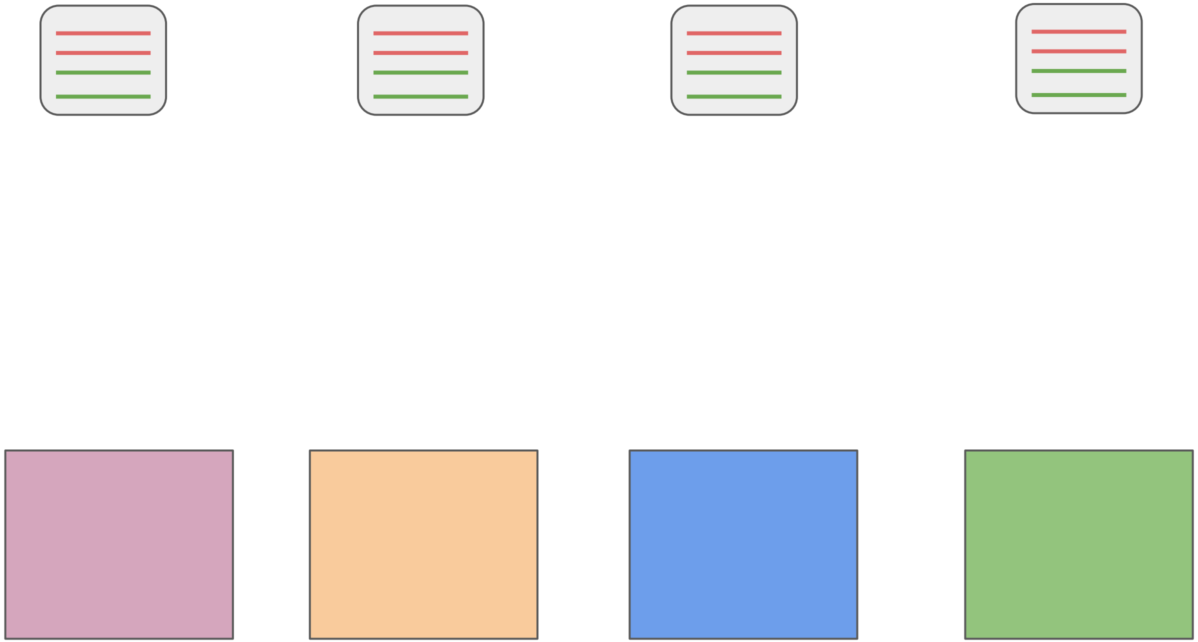
Batching multiple datasets

This is particularly simple for Graph Neural Networks



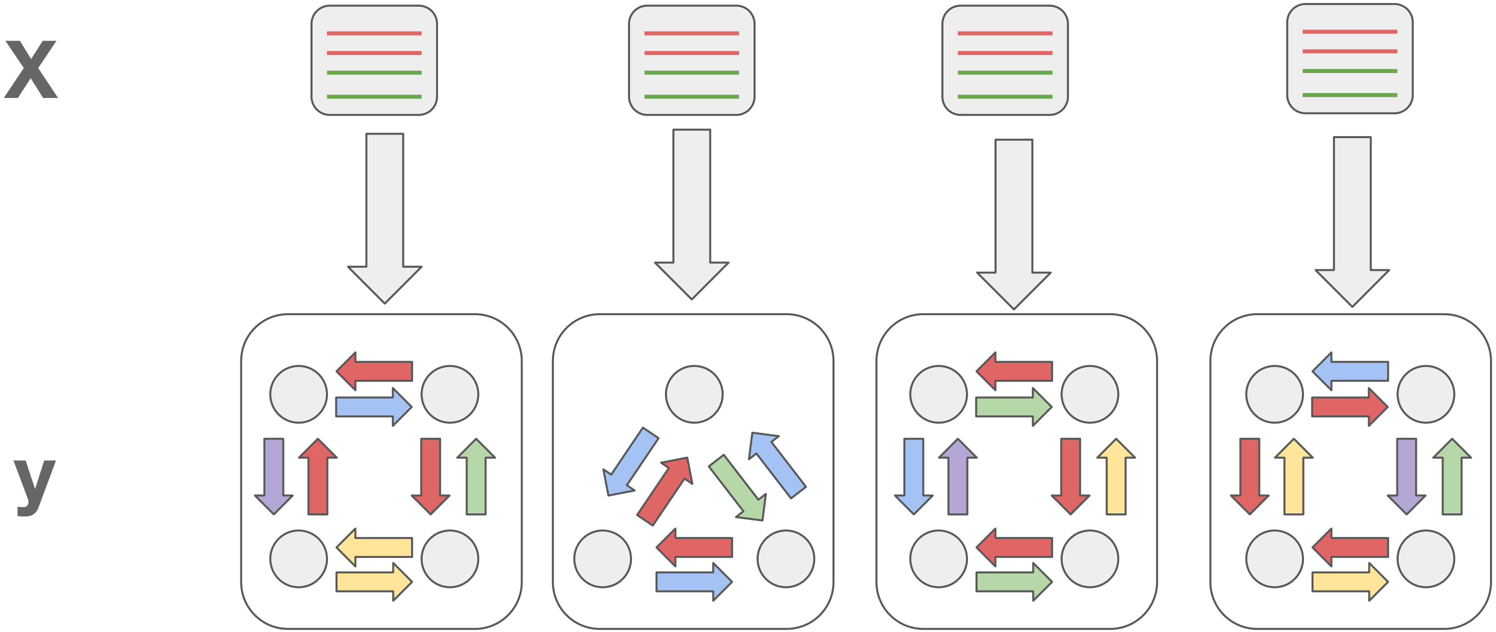
Batching multiple datasets

This cannot be done in non-modular meta-learning algorithms



Learning the proposal function

Create a dataset from meta-training information

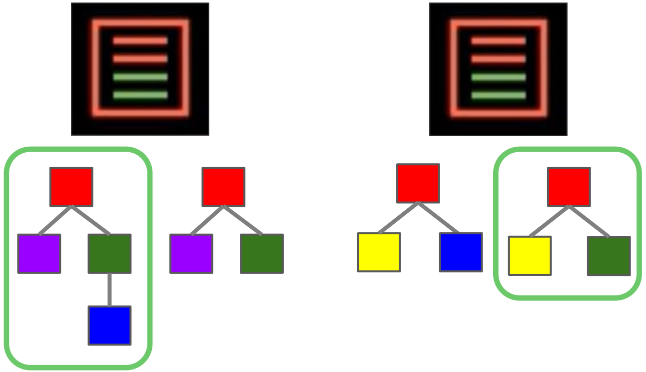


We're simultaneously learning to learn and learning to optimize

Learning the proposal function

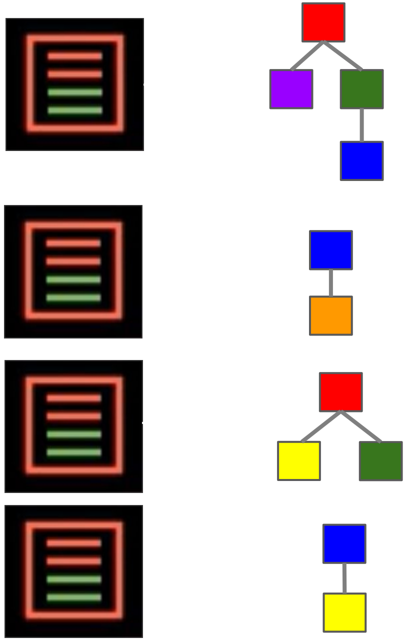
Slow

Simulated annealing with learned proposal function



Fast

Proposal function imitates simulated annealing



Self-learning modular meta-learning

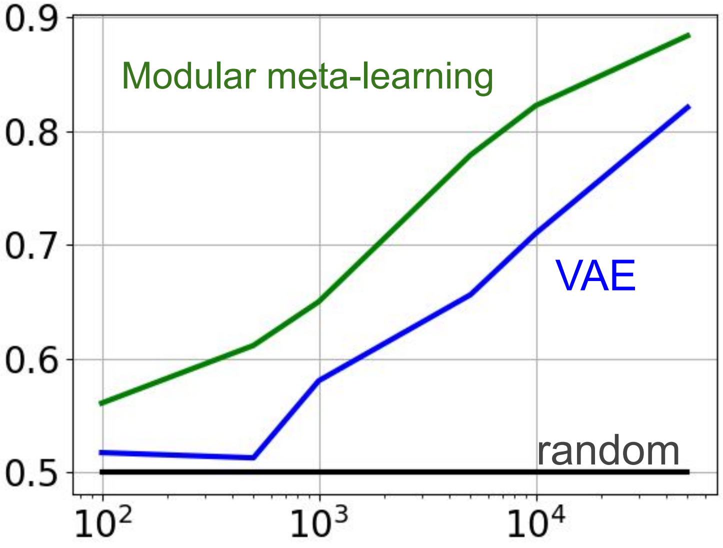
- Random proposal function: slow for 2^20 search space
- Bottom up proposal: trajectories → structure
    - doesn't require good current structures
    - still uniform prior over structures
- Top-down proposal: structure→ structure
    - requires good structures to work
    - can form complex prior over structures

Improved Results

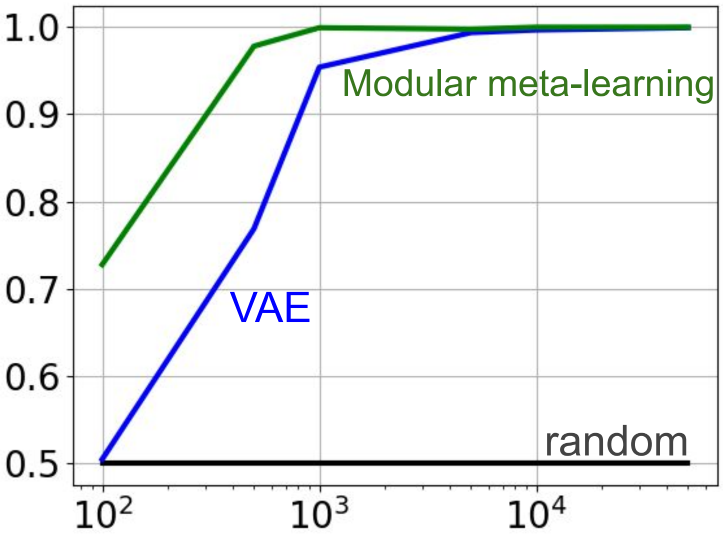
	Springs		Charged	
Prediction steps	1	10	1	10
Static	7.93e-5	7.59e-3	5.09e-3	2.26e-2
LSTM(single)	2.27e-6	4.69e-4	2.71e-3	7.05e-3
LSTM(joint)	4.13e-8	2.19e-5	1.68e-3	6.45e-3
NRI (full graph)	1.66e-5	1.64e-3	**1.09e-3**	3.78e-3
(Kipf et al., 2018)	**3.12e-8**	**3.29e-6**	**1.05e-3**	3.21e-3
Modular meta-l.	**3.13e-8**	**3.25e-6**	**1.03e-3**	**3.11e-3**
NRI (true graph)	1.69e-11	1.32e-9	1.04e-3	3.03e-3

Model	Springs	Charged
Correlation(data)	52.4	55.8
Correlation(LSTM)	52.7	54.2
(Kipf et al., 2018)	**99.9**	82.1
Modular meta-l.	**99.9**	**88.4**
Supervised	99.9	95.0

Model-based approach leads to data efficiency

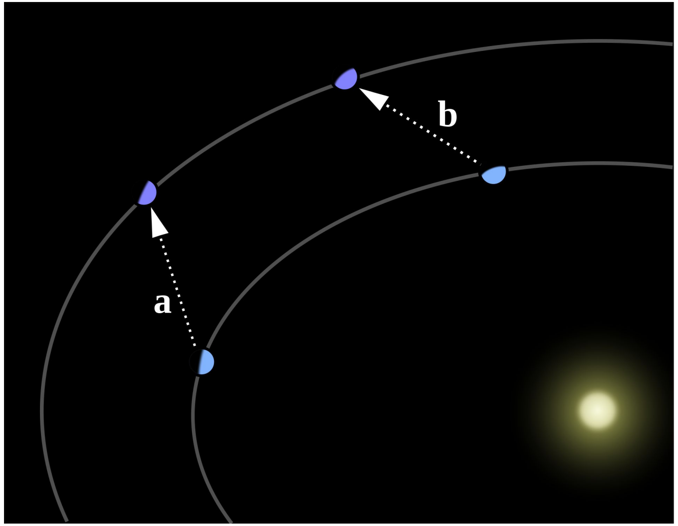


Charged



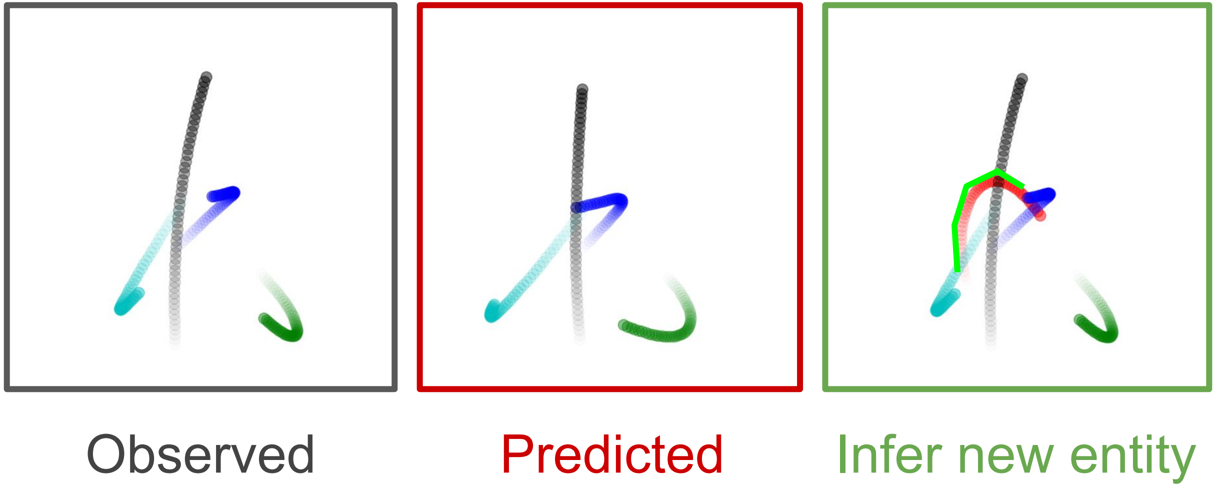
Springs

Reasoning about our own knowledge



Neptune affecting Uranus orbit

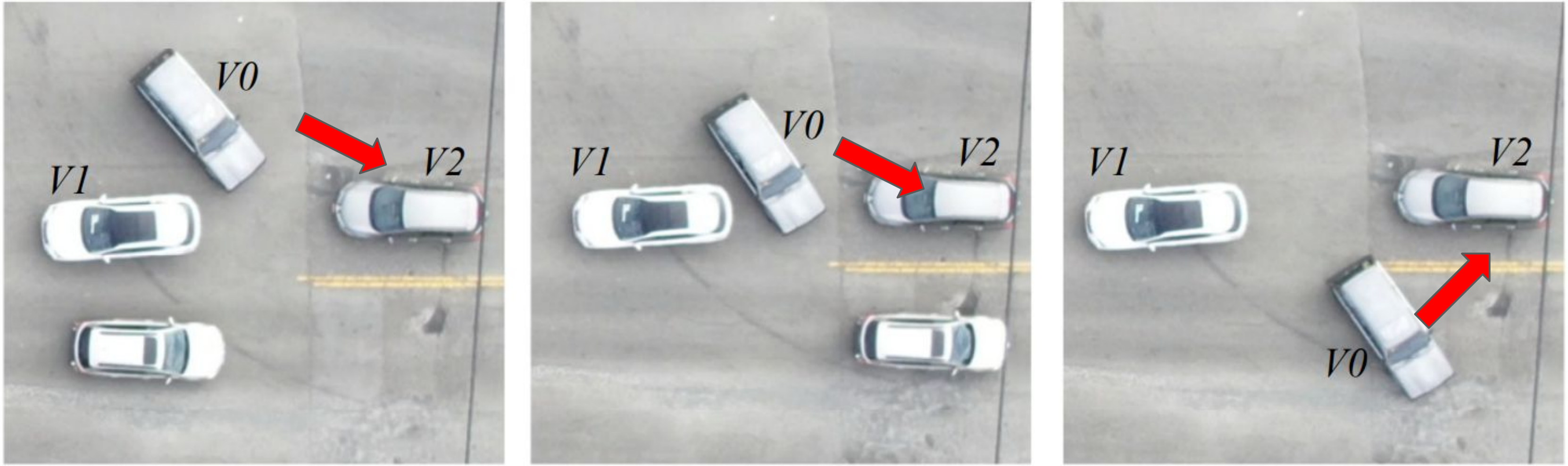
Reasoning about our own knowledge



Found missing node with precision comparable to some baselines which had the state of the particle up to 10 steps before

Real life application: self-driving cars

Understanding the intentions of other drivers
is one of the major roadblocks (pun intended) for training
self-driving cars

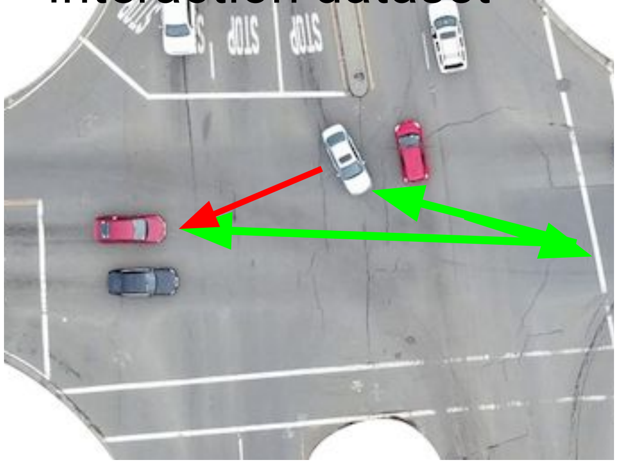


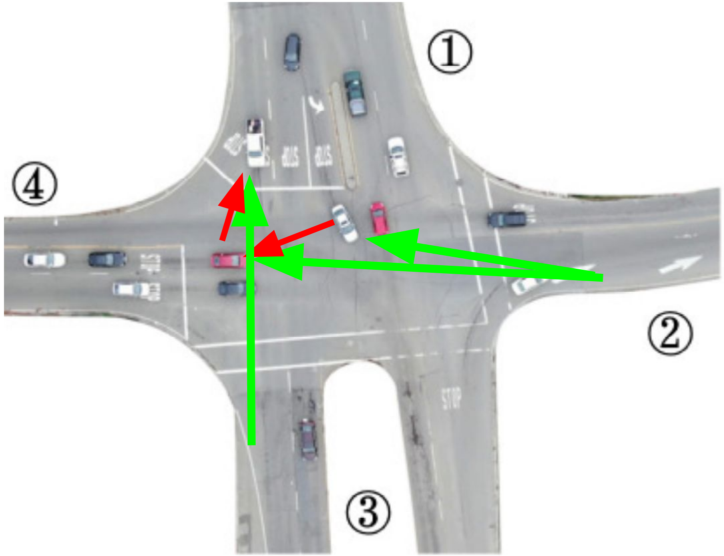
$t = 0$ $t = 1$ s $t = 3.2$ s

Real life application: self-driving cars

Understanding the intentions of other drivers
is one of the major roadblocks (pun intended) for training
self-driving cars

Interaction dataset





Summary

- Model-based approach to NRI is much more data efficient
- and can tackle problems for which it was not trained
- We can use information collected during meta-training to learn to optimize the structure search (i.e. what was accepted, what was rejected during simulated annealing)
- Modular meta-learning can scale to much larger meta-datasets