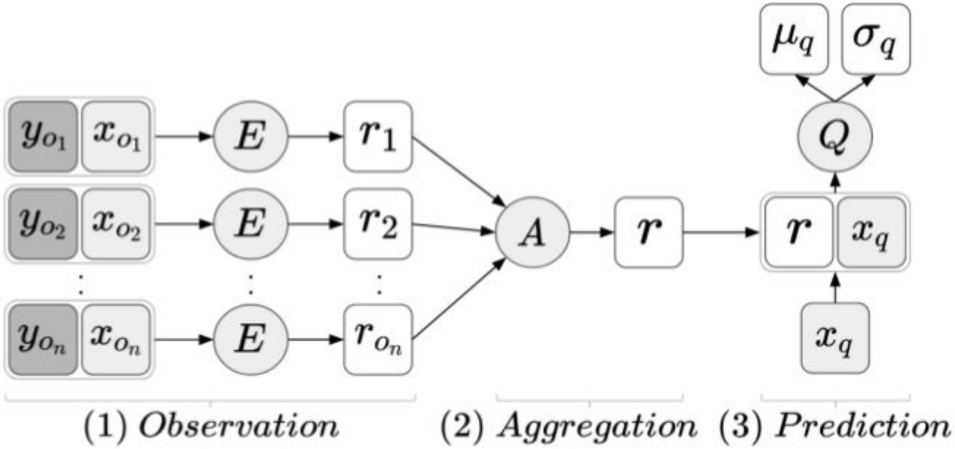
Adaptive Conditional Neural Movement Primitives via Representation Sharing Between Supervised and Reinforcement Learning

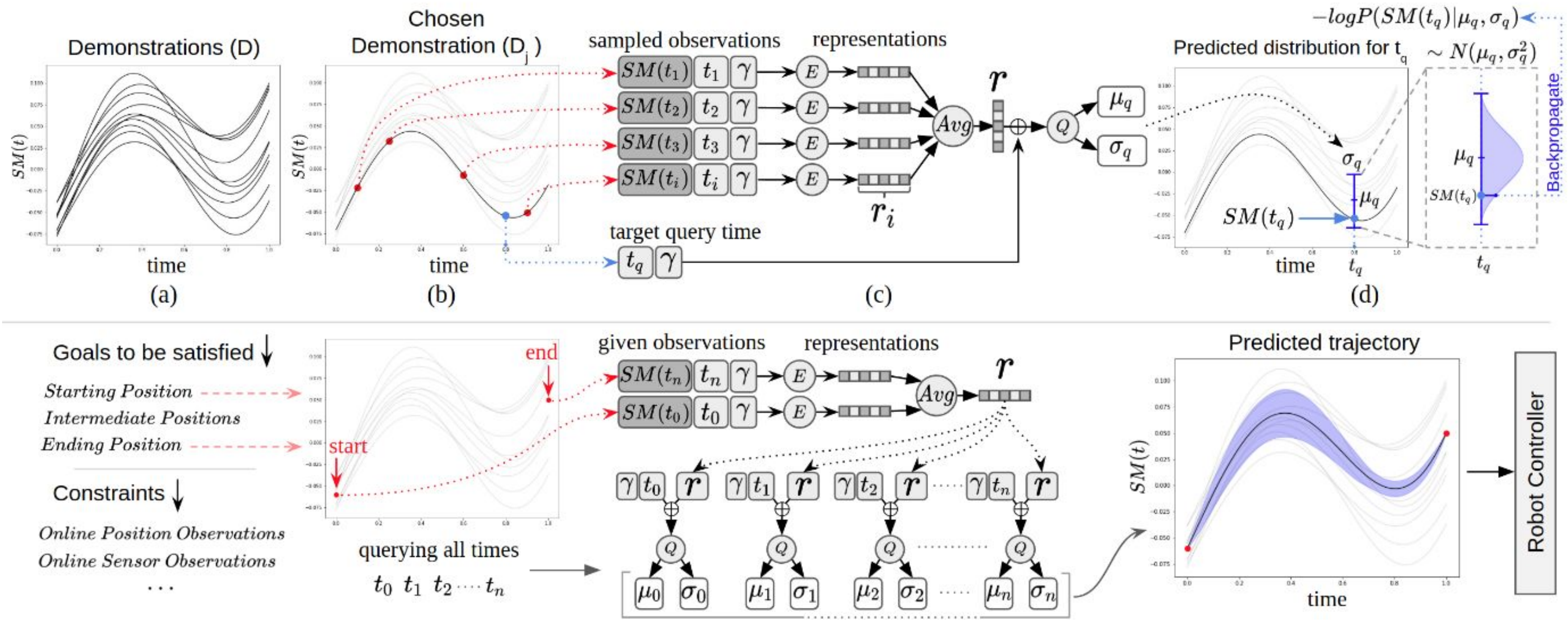
M.Tuluhan Akbulut, M.Yunus Seker, Ahmet E. Tekden, Yukie Nagai, Erhan Oztop, Emre Ugur

Conditional Neural Processes

- *Conditional Neural Processes,* Garnelo et.al. 2018



Motor primitives based on CNP



Problem and solution

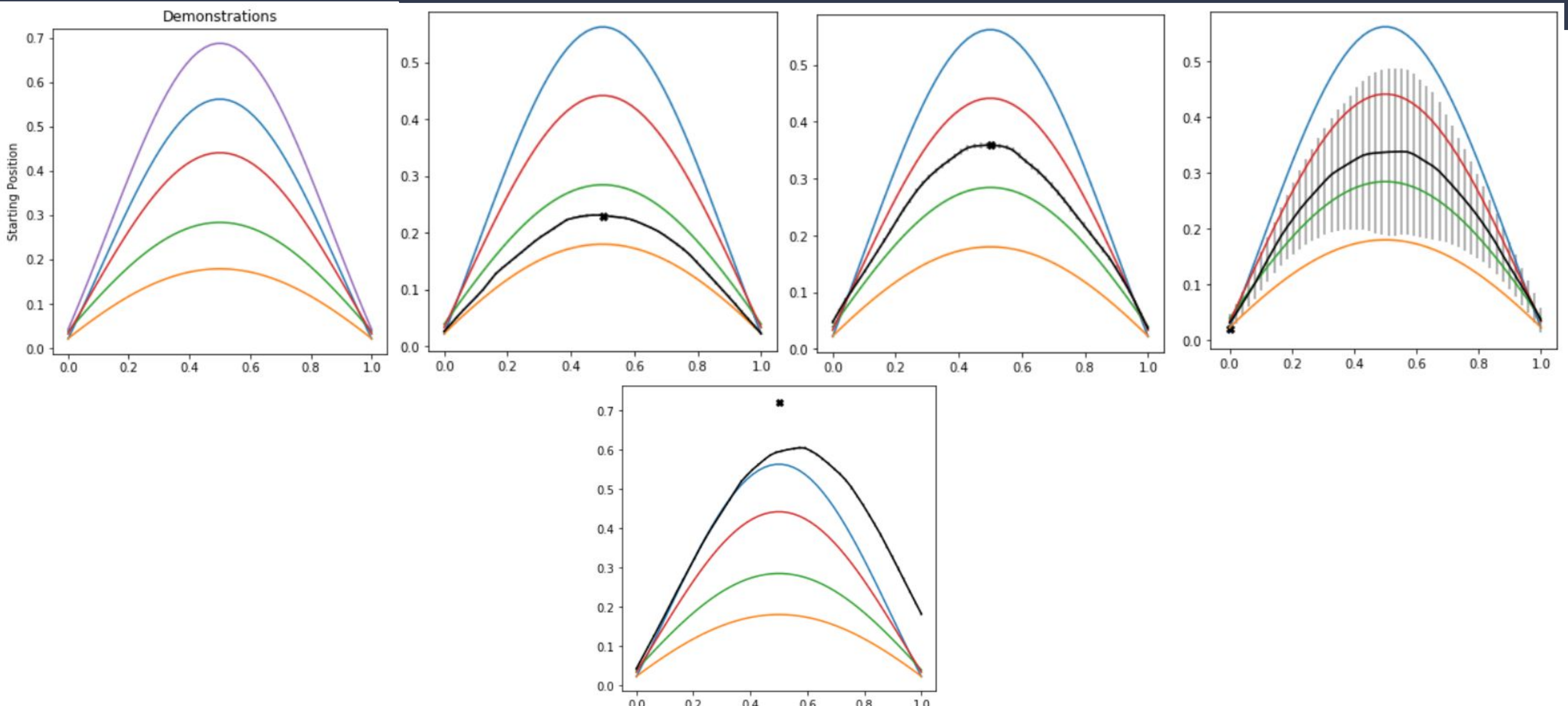
Problems:

- Neural network
- Capabilities are limited with data
- Can’t do extrapolation

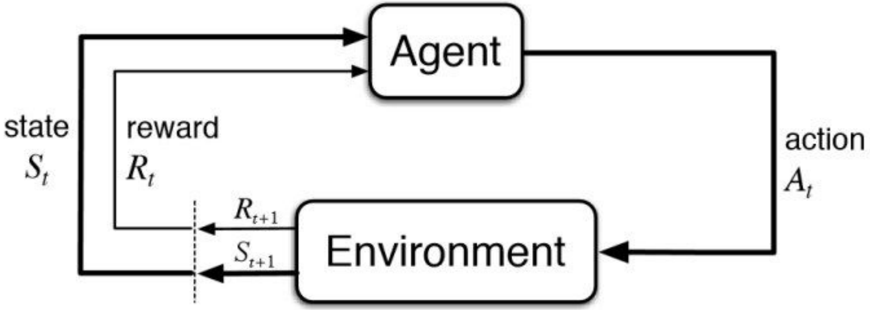
Solution:

- Self-error detection
- Adaptation algorithm after error
- New task constraints are reached by RL

Capabilities of CNMP



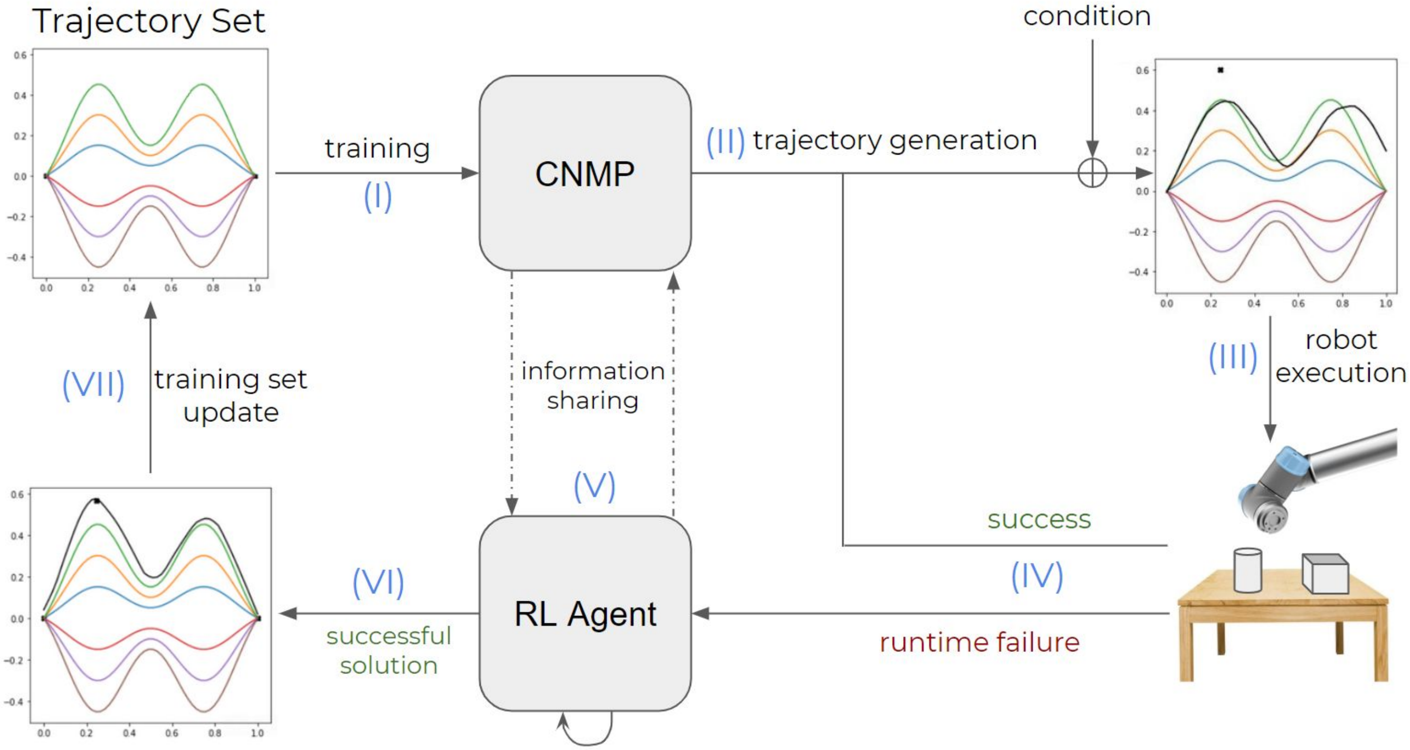
What is RL?



$$\pi_\theta(\mathbf{u}_t|\mathrm{x}_t) \rightarrow \tau = \{\mathbf{x}_1, \mathbf{u}_1, \mathbf{x}_2, \mathbf{u}_2, \ldots, \mathbf{x}_T, \mathbf{u}_T\}$$

$$\max_{\pi_\theta} E_{\tau\sim\pi_\theta}\left[\sum_t \gamma^t\, r(\mathrm{x}_t, \mathbf{u}_t)\right]$$

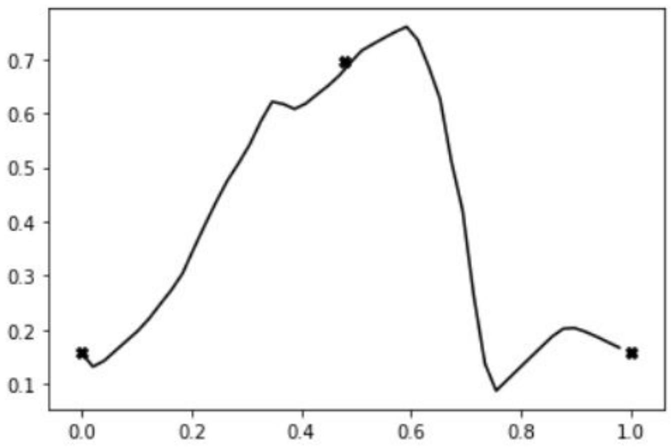
Method



Trajectory Set
training
(I)
CNMP
(II) trajectory generation
condition
(III)
robot execution
(VII)
training set update
information sharing
(V)
success
(IV)
(VI)
RL Agent
successful solution
runtime failure

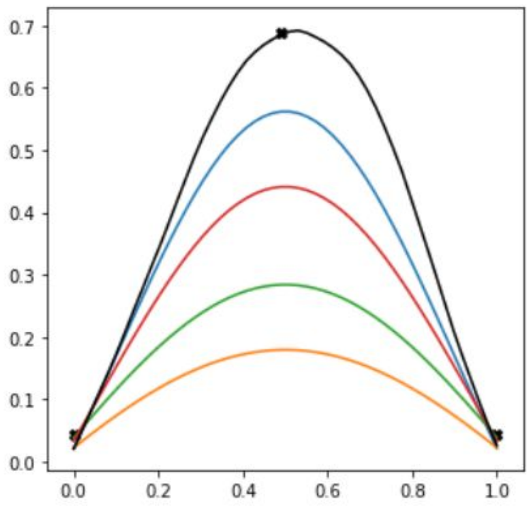
Correction

- Only reinforcement learning:



- New trajectory should look like expert trajectory(be optimal) to be added CNMP framework.

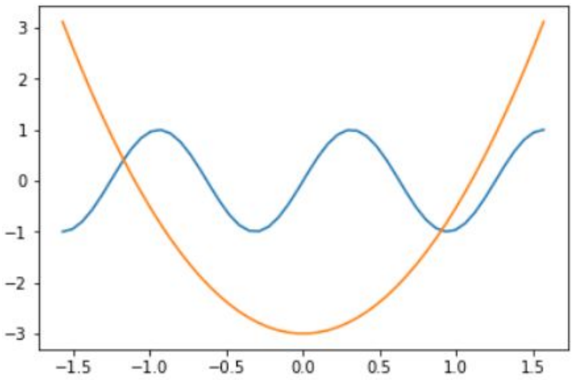
- RL agent trained on all trajectories (mixed data):

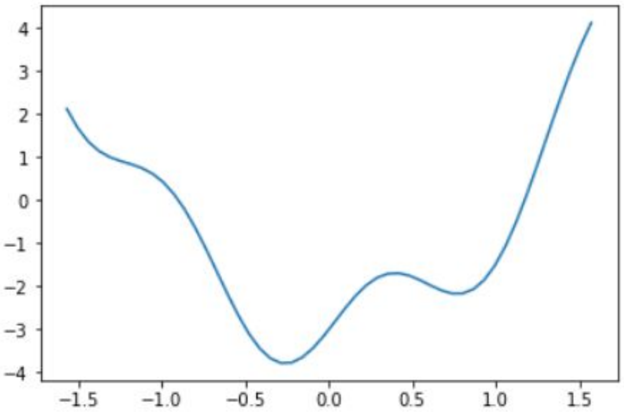


My approach:

policy gradient: $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

maximum likelihood: $$\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)$$





Literature

- How to combine RL and supervised learning:

  Most algorithms:

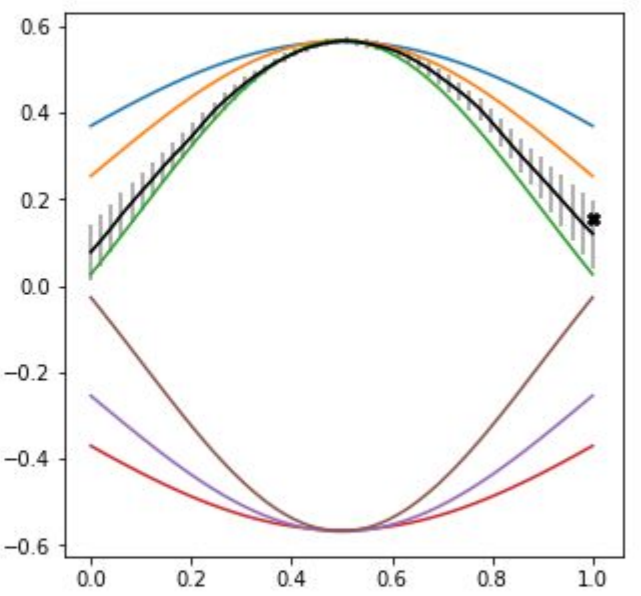
  Adding expert demonstrations to replay buffer:

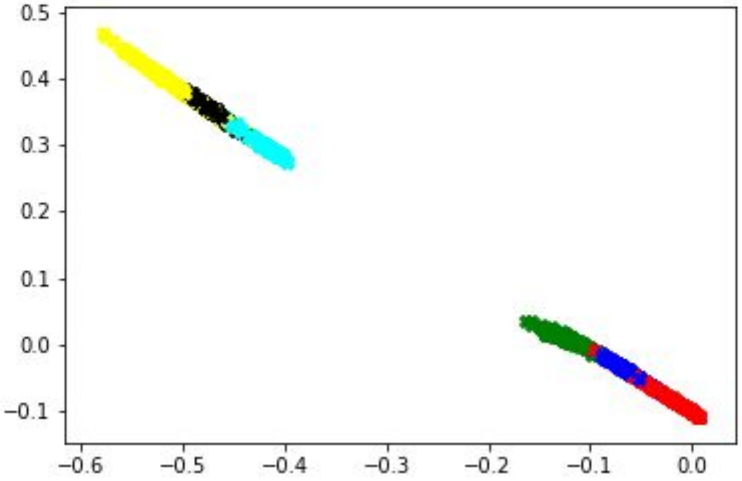
  Deep Q-Learning by Demonstrations by *Hester et.al. Google Deepmind ,2018*

$$J_E(Q) = \max_{a \in A}[Q(s,a) + l(a_E, a)] - Q(s, a_E)$$

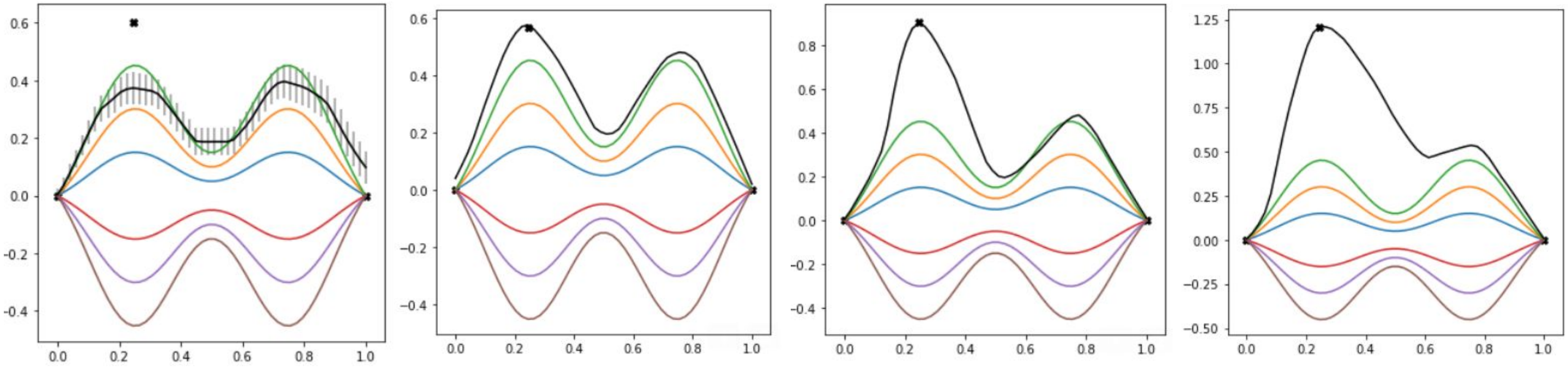
$l(a_E, a)]$ = 0 when $a = a_E$

Latent space visualization

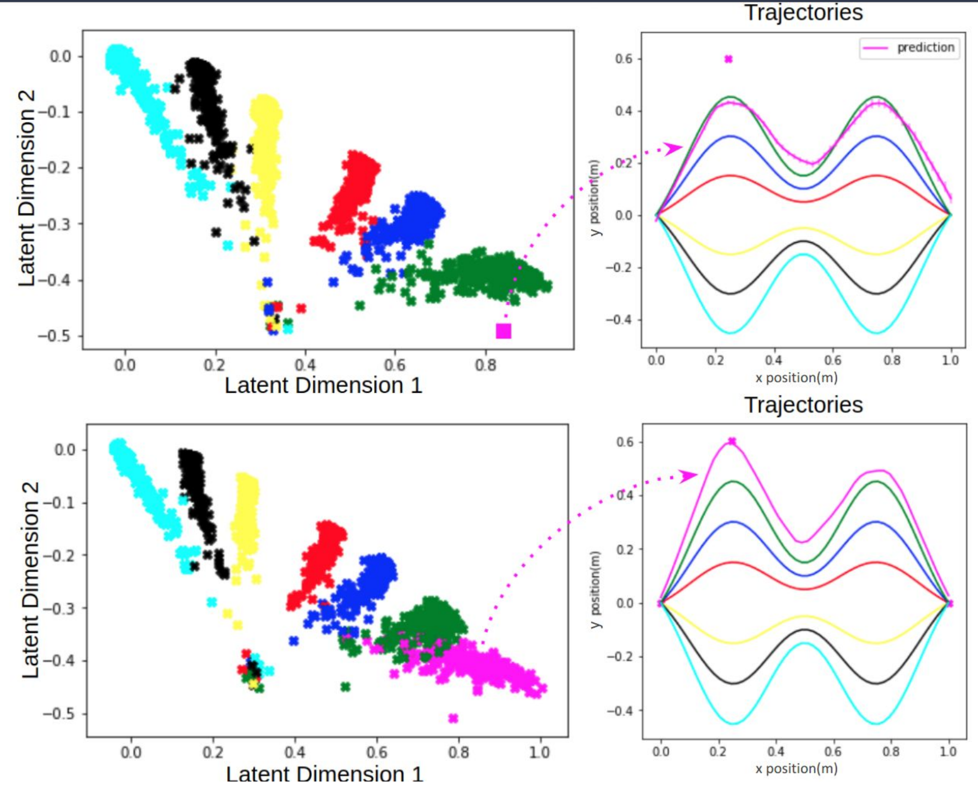




Results with a more complex task:

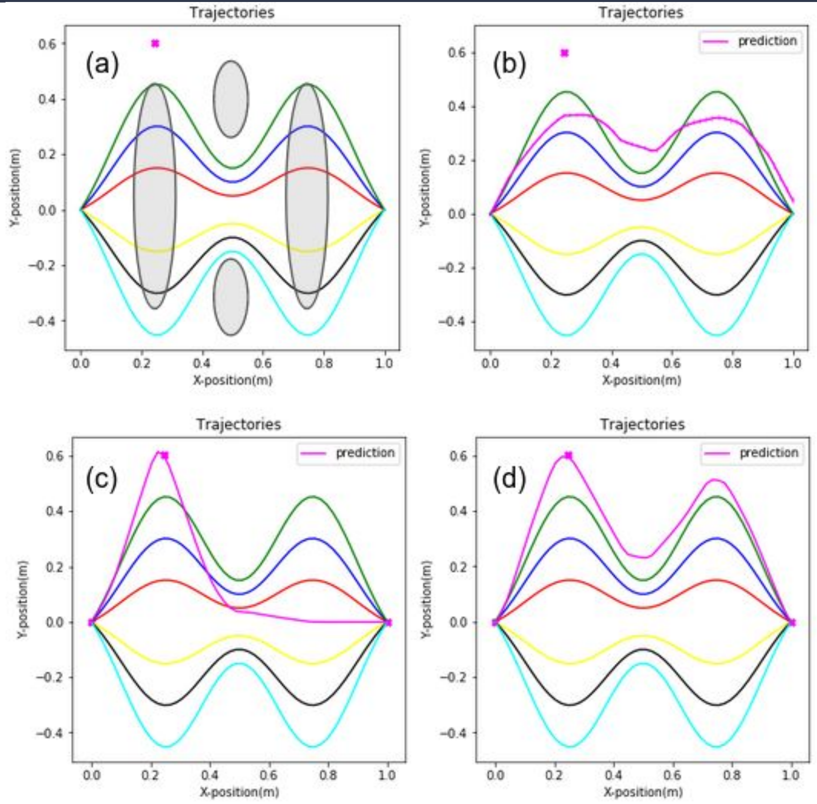


Latent space visualization

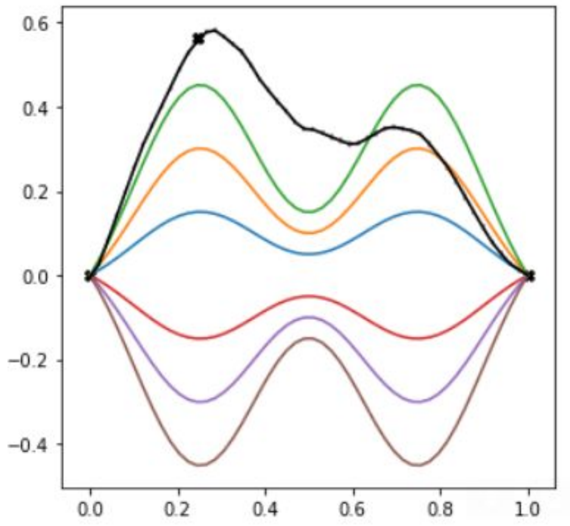


Why do we train together

- The policy agent is stochastic
- The gradient is stochastic
- There is no guarantee to preserve the policy shape:



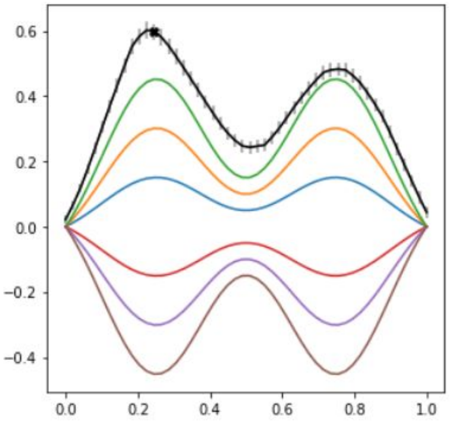
Is it same with regression:

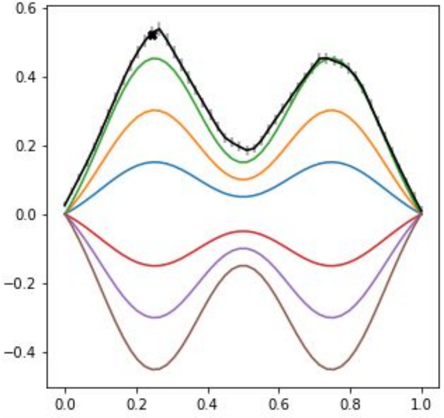


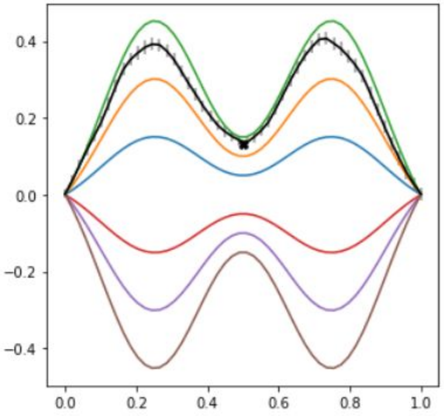
Produced gradients only for 3 points.

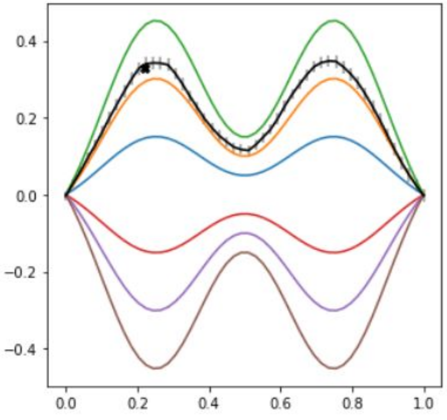
In my setup, we let gradient to backpropagate for all time-steps.

Closing the loop:









Experiments:

- Video

Disadvantages

- Since we are training together, it becomes harder to converge(tradeoff between expert trajectory and RL trajectory)

- Harder to differentiate from CNMP

- Takes more iterations, needs optimization.

- Needs further analysis about convergence and representation map

- Oscillation


Thank You
For Your Atten