



IMITATION META LEARNING

Learning to Learn | An Introduction

When you teach a child something you take away, you take away chance of discovering it for himself.

--Jean Piaget

Presentation By:
Tejas Gaikwad (MT19AI021)
MTech- Artificial Intelligence



Reference Papers for this presentation

WATCH , TRY , LEARN : META-LEARNING FROM DEMONSTRATIONS AND REWARDS

(ICLR 2020)

Authors

Allan Zhou, Eric Jang, Daniel Kappler, Alex Herzog, Mohi Khansari, Paul Wohlhart, Yunfei Bai, Mrinal Kalakrishnan, Sergey Levine, Chelsea Finn

ONE-SHOT IMITATION FROM OBSERVING HUMANS VIA DOMAIN-ADAPTIVE META-LEARNING

(ICLR 2018)

Authors

Tianhe Yu, Chelsea Finn*, Annie Xie, Sudeep Dasari, Tianhao Zhang,
Pieter Abbeel, Sergey Levine*

Video on Imitation Learning in Humans



Credits: youtube



Imitation Learning

Definition

Imitation learning techniques aim to mimic human behavior in a given task. An agent (a **learning** machine) is trained to perform a task from demonstrations by **learning** a mapping between observations and actions.



Meta-Learning

Definition

Meta learning is a subfield of machine learning where automatic learning algorithms are applied to metadata about machine learning experiments.

(Example of Reading books/
Research Papers)

Metadata is "data that provides information about other data". In other words, it is "data about data". Many distinct types of metadata exist, including descriptive metadata, structural metadata, administrative metadata, reference metadata and statistical metadata.

Credits: Wikipedia

Index



- Problem Statement
- Goal
- Literature Survey
- Proposed Solution
- Implementation
 - Pre - requisites
 - Agent Design (Phase I and Phase II)
 - Loss Functions
 - Algorithms for Phase I and Phase II
 - WTL Implementation (for Phase I and Phase II) Model Description
 - Visual Model
 - Actor Model
- Experiments / Results
- Future Work
- Conclusion



Problem Statement

Meta-Learning
(Learn to Learn)

Learning a complex
vision-based task with
very few
demonstrations



Goal

Meta-Learning
(Learn to Learn)

Using Meta-Learning approach that can learn from a single demonstration and subsequent reinforcement learning trails

Literature Survey



Related Work

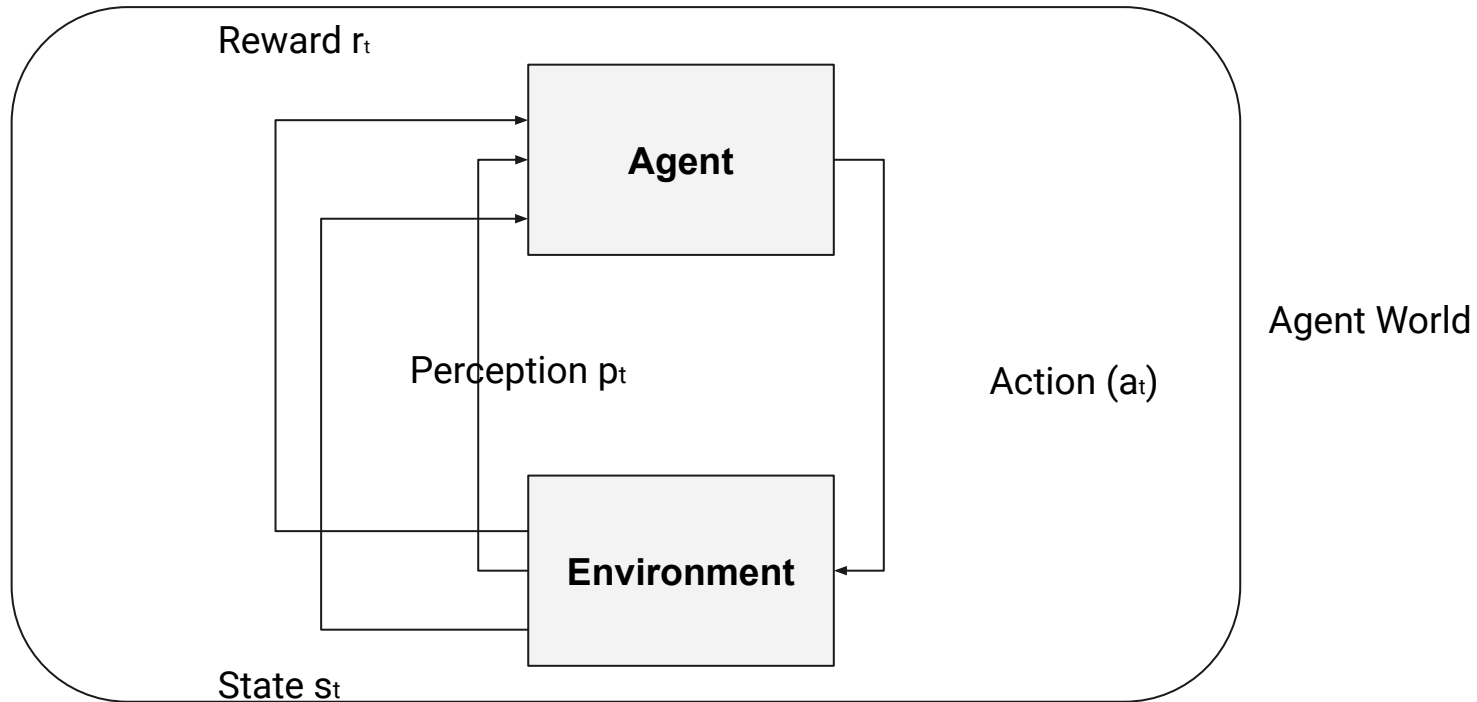
Meta-Learning
(Learn to Learn)

- **Meta-Learning / Learning to learn** (Thrun & Pratt, 1998; Schmidhuber, 1987; Bengio et al., 1992; Hochreiter et al., 2001)
- **Meta-Imitation Learning** (Duan et al., 2017; Finn et al., 2017b; James et al., 2018; Paine et al., 2018)
- **Meta-Reinforcement Learning** (Duan et al., 2016; Wang et al., 2016; Mishra et al., 2018; Rakelly et al., 2019)
- **Few-shot imitation learning** (Duan et al., 2017; Finn et al., 2017b; Yu et al., 2018; James et al., 2018; Paine et al., 2018)
- **Multi-task and meta-reinforcement learning** (Duan et al., 2016; Wang et al., 2016; Finn et al., 2017a; Mishra et al., 2018; Houthoofd et al., 2018; Sung et al., 2017; Nagabandi et al., 2019; Sæmundsson et al., 2018; Hausman et al., 2017)
- **Demonstration to address the meta-exploration problem** (Gupta et al., 2018; Stadie et al., 2018).
- **Meta-Learning with thousands of iterations** (Duan et al., 2016; Wang et al., 2016; Finn et al., 2017a; Mishra et al., 2018)

Implementation

Pre - requisites

The Bigger Picture



Some Keywords



Agent Representation = AG

Action by agent at instance t = a_t

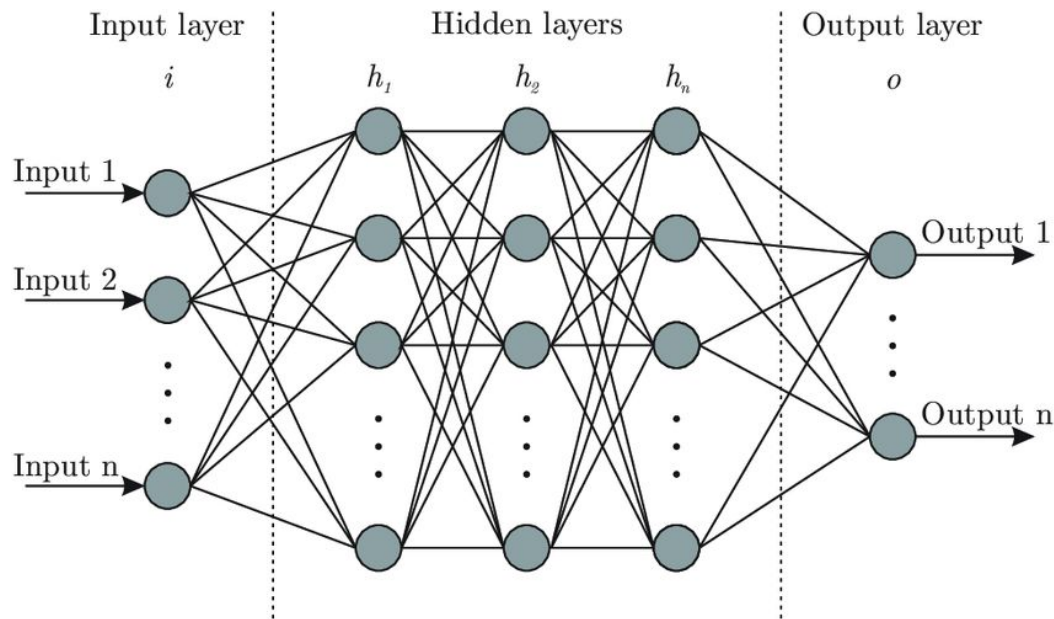
Action a_t = $f(\text{perception through sensors, present state, Previous reward})$

Reward r_t = $f(\text{action taken, State Achieved})$

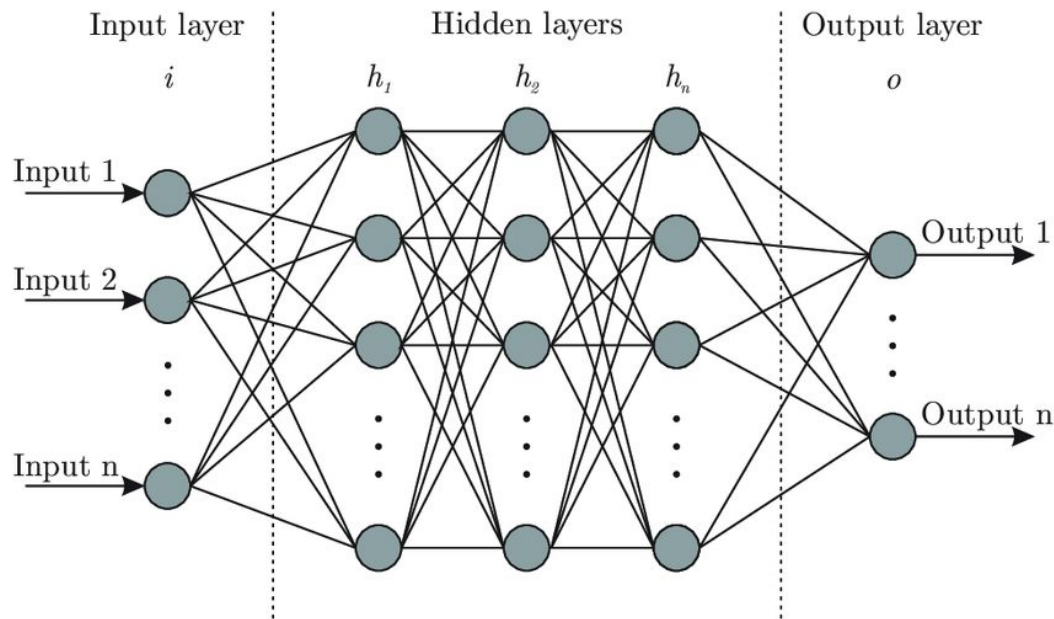
Set of States = $S = \{s_0, s_1, \dots\}$

Set of Actions taken = $A = \{a_0, a_1, a_2, \dots\}$

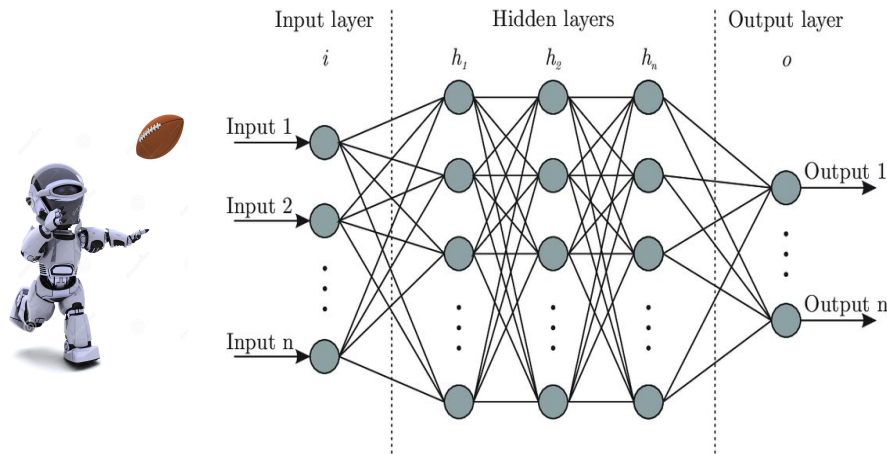
Imitation Learning using Neural Networks



Imitation Learning using Neural Networks



Imitation Learning using Neural Networks



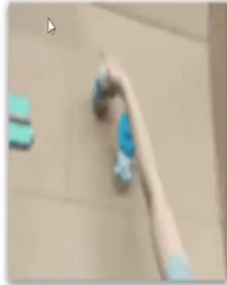
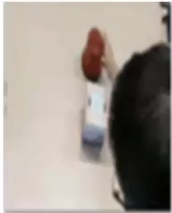
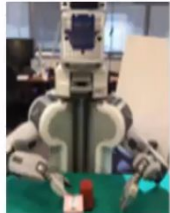


Problems?

- Lot of Demonstration
- Different objects?
- Learning Embeddings

How meta-learning can help?

- Use Prior knowledge build-up from many other tasks (Making agent learn, how to learn tasks)
- Then learning new task with just a single demonstration (One Shot)

	Dataset	D^h	D^r	One-Shot Learning
Task 1				
Push left the pack				
Task 2				
Push left the cup				Push left the bottle

T. Yu et. al. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning, 2018



Problem Definition

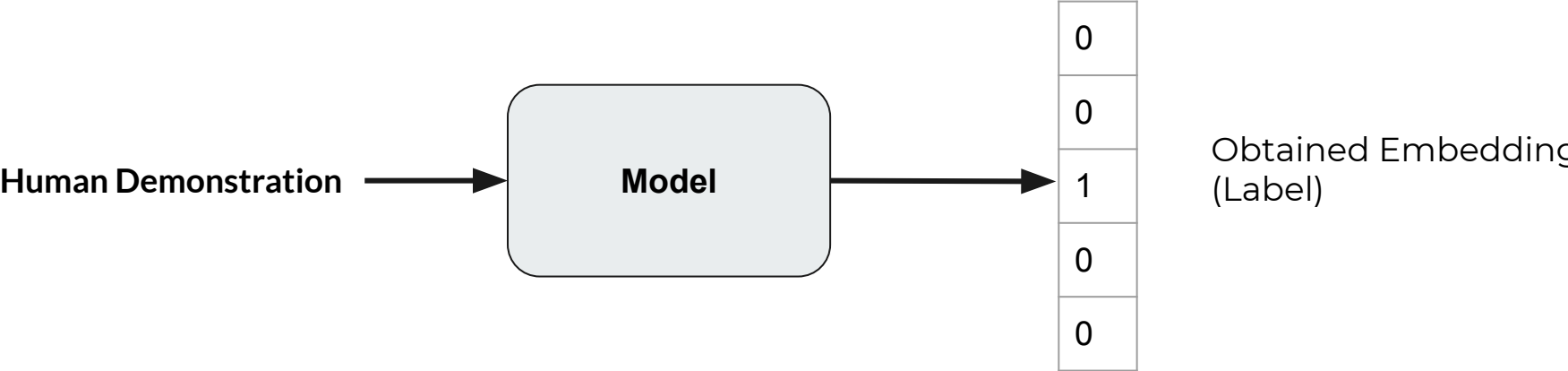
- So, there are set of tasks, let say $T = \{T_0, T_1, \dots\}$ sampled from $p(T)$ distribution
- Demonstrations (Human and robotic) $= \{D_h, D_r\}$
- Demonstration from Human will give set of observations $= \{o_0, o_1, o_2, \dots\}$
- Demonstration from robots(agent) will give set of observations, action and state $= \{o_0, a_0, s_0, o_1, a_1, s_1, \dots\}$
- Example :
 - States : Joint angles
 - Actions :
 - Torque applied to Joints
 - Gripper



Problem Definition

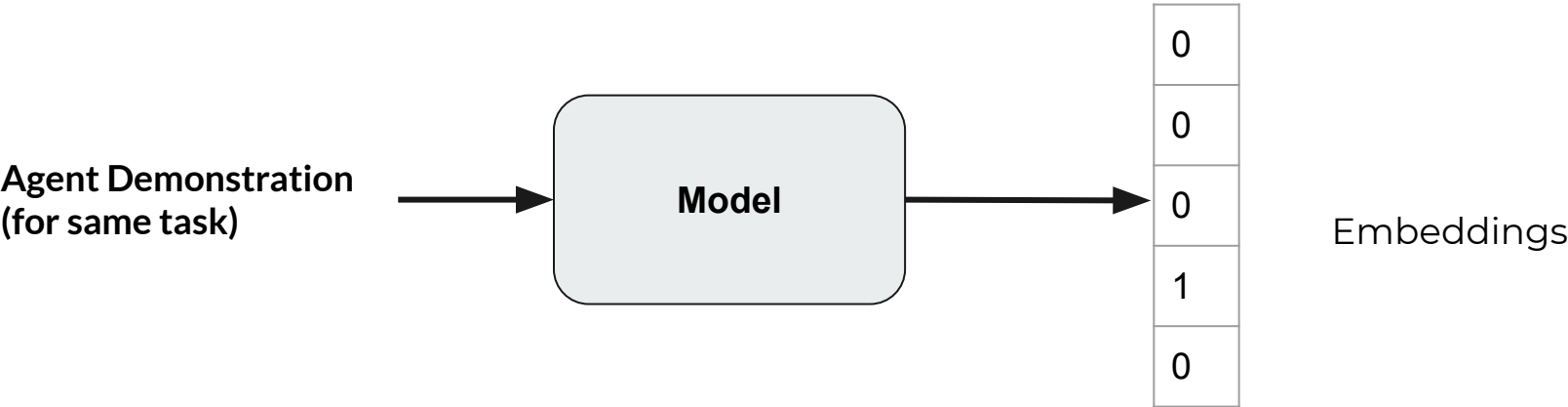
- **Policy :** A policy is a mapping from perceived states of the environment to actions to be taken when in those states
 - $\Pi_{\theta} \Rightarrow$ Policy with theta as set of all trainable parameters $\Rightarrow P(\text{action}/\text{states, observations})$
 - Output of this trained policy would be demonstration by Agent
- **Loss Function:**
 - $\mathcal{L}(\theta, \{\text{actions, state, observations}\}) = \sum (\text{over task } t \text{ belongs to } T) \log(\Pi_{\theta}(a_t/s_t, o_t))$
- **Optimisation Function:**
 - Minimising the above loss Function $\Rightarrow \min \sum (\text{over } T \text{ sampled from distribution } p(T)) \sum (\text{human demonstration}) \sum (\text{agent demonstration}) \log(\Pi_{\theta}(a_t/s_t, o_t))$

Training of the Network (Phase I)





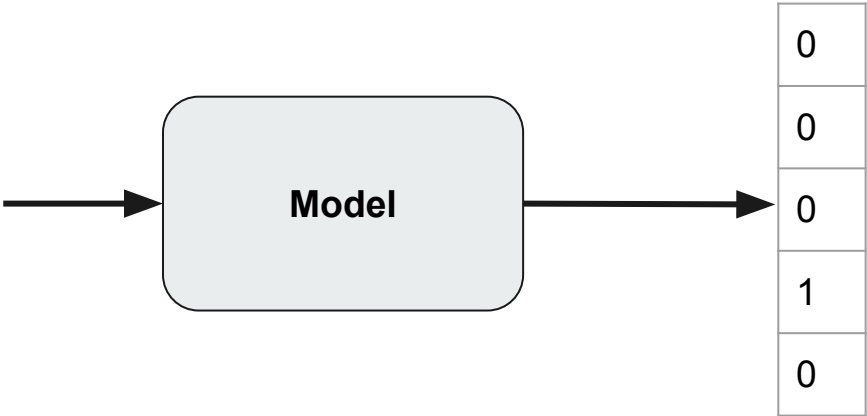
Training of the Network (Phase I)





Training of the Network (Phase I)

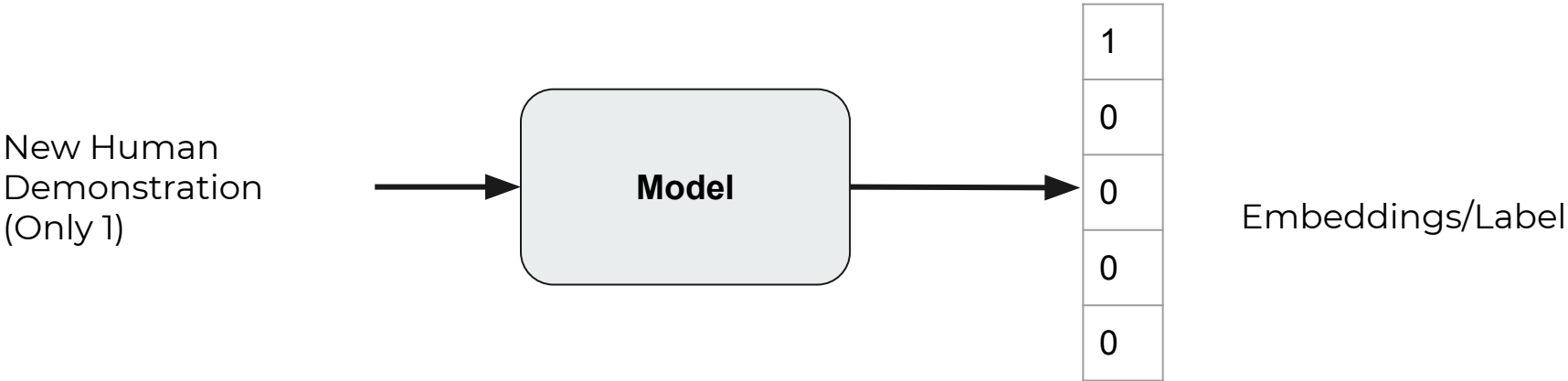
Similarly it goes,
First Human
Demonstration for a
particular task then
Agent
Demonstration for
the same task



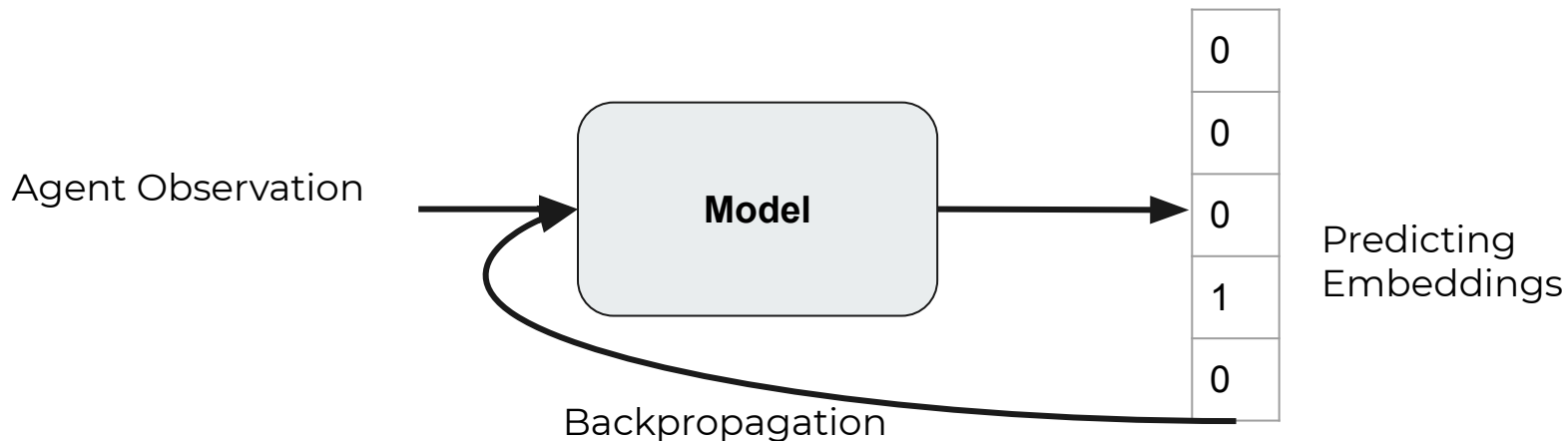
Embeddings



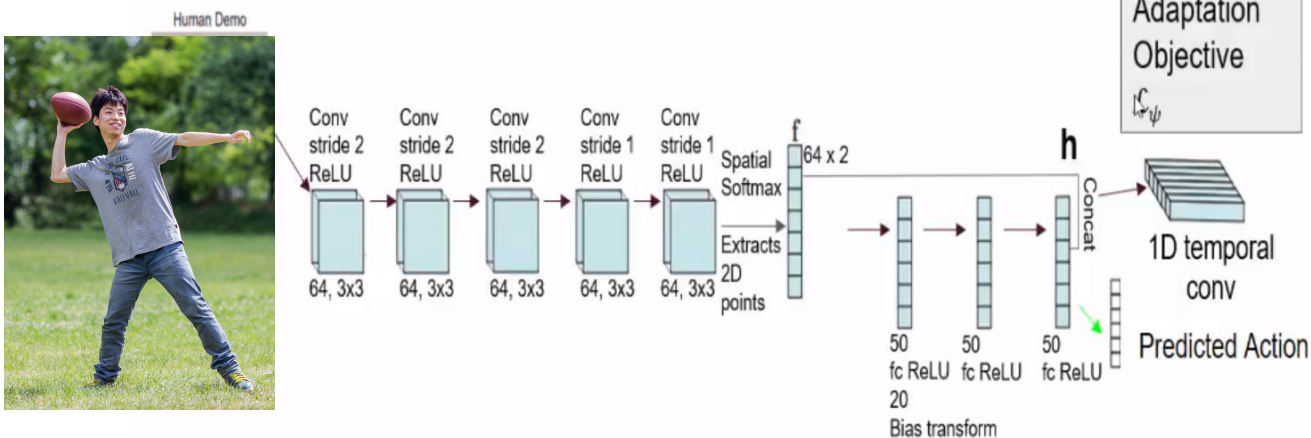
Testing of the Network (Phase II)



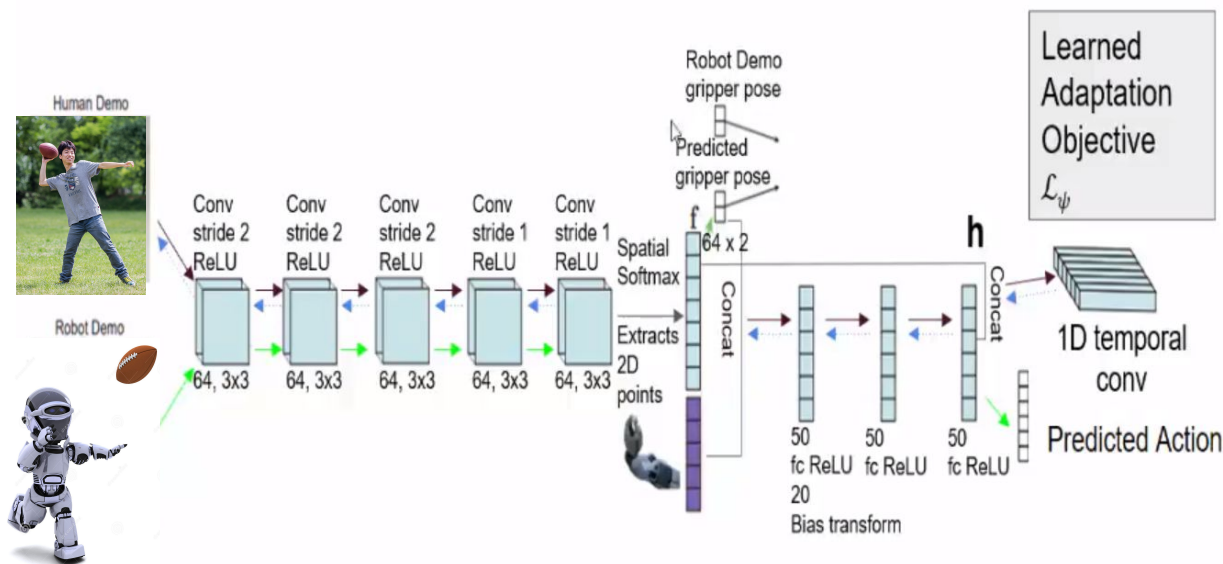
Testing of the Network (Phase II)



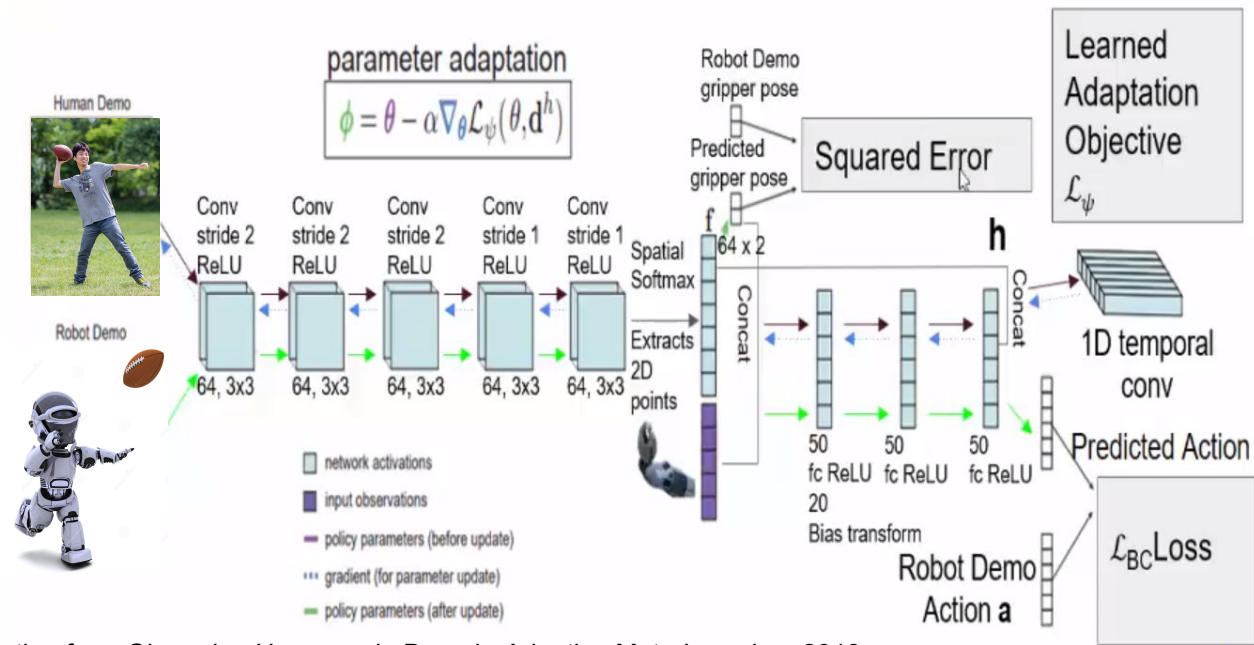
Policy Network



Policy Network



Policy Network



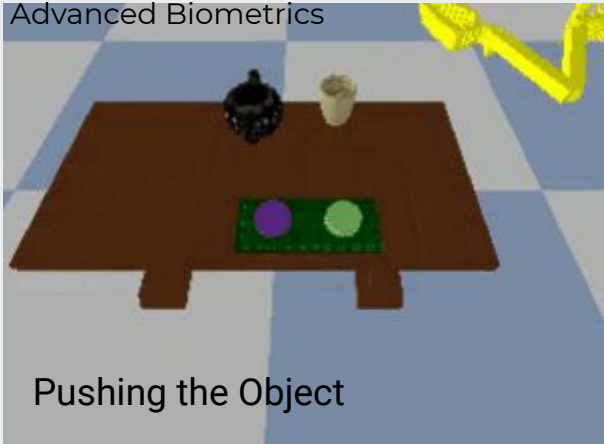
Implementation

Agent Design Phase I and Phase II



What Agent is trying to learn

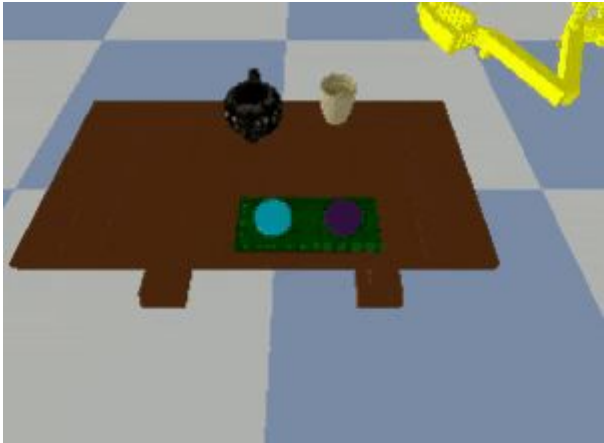
- With few/single demonstrations for the tasks like pushing, grabbing, uplifting objects making agent learn the policies itself
- Imitating the action with few/single demonstration



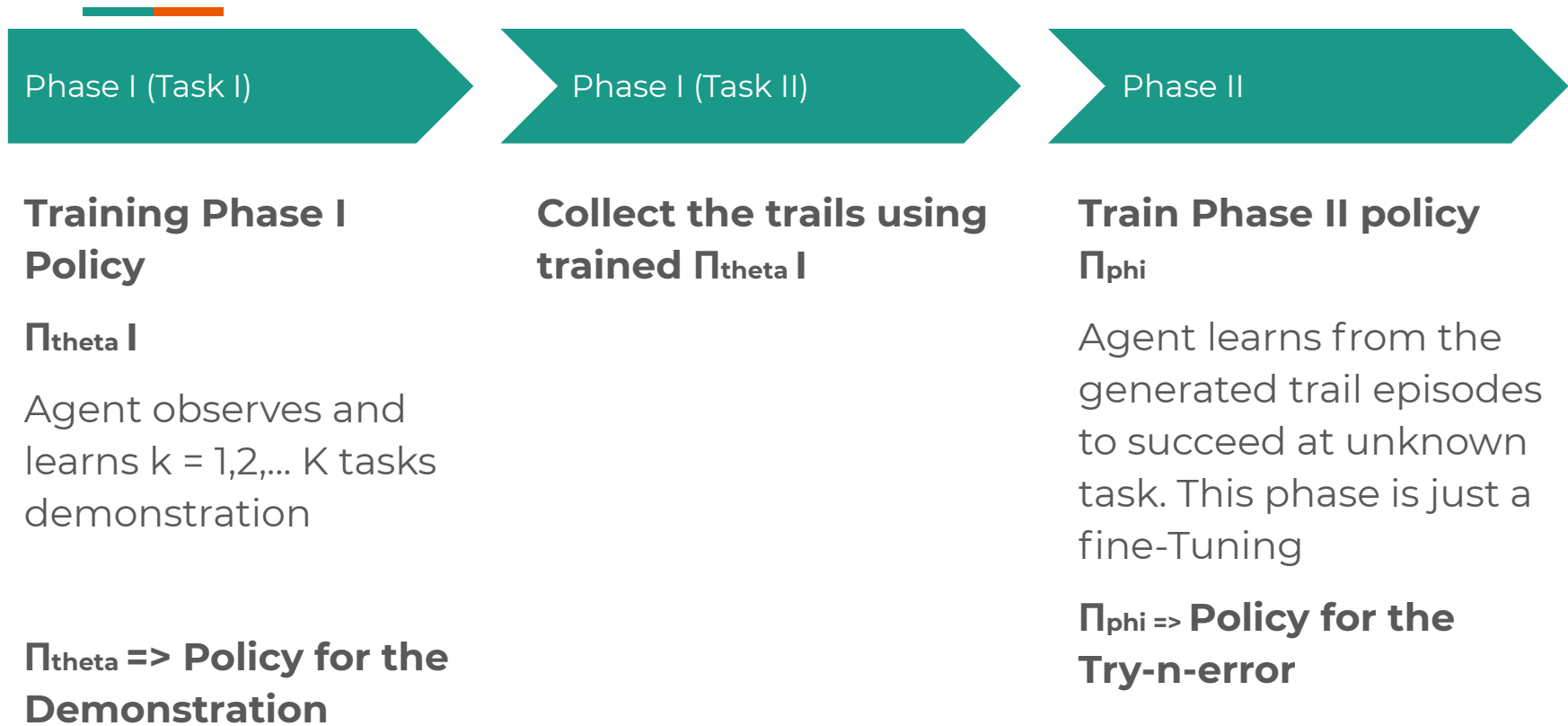
Pushing the Object



Pushing the Button



Picking the Object



- A Trail episode is represented as ' τ ' and given as ' $\tau = \{s_t, a_t, r_t(s_t, a_t)\}$ '
- Demonstration represented as $\mathbf{d} = \{s_t, a_t\}$
- Learning to Imitate and Try-Error Method => **Demonstration + try-n-error experiment**
 - Meta-Learning Phase I policy is suitable for gathering the information about a task given demonstration.
 - Meta-Learning Phase II learns from both demonstrations and trails produced by Phase I policy.
 - Policies:
 - **Phase I policy** => $\Pi_{\theta}(\text{action/state, demonstration})$, where θ is all the learnable parameters (Weights)
 - **Phase II policy** => $\Pi_{\phi}(\text{action/state, demonstration, try-n-error})$, here ϕ represents learnable parameters (weights)
 - First Phase I policy is trained and then freezed.
 - Then, collection of trail data is done for each meta-learning task τ_i
 - Lastly Phase II policy is trained using these trail data
- Phase I must be trained in such manner that it will provide useful exploration for inferring task. *Thompson's Sampling is used for this*

Implementation

Loss Functions

Phase I loss for Task \mathcal{T}_i

$$\mathcal{L}^I(\theta, \mathcal{D}_i^*) = \mathbb{E}_{\{\mathbf{d}_{i,k}\} \sim \mathcal{D}_i^*} \mathbb{E}_{\mathbf{d}_i^{\text{test}} \sim \mathcal{D}_i^* \setminus \{\mathbf{d}_{i,k}\}} \mathbb{E}_{(s_t, a_t) \sim \mathbf{d}_i^{\text{test}}} [-\log \pi_{\theta}^I(a_t | s_t, \{\mathbf{d}_{i,k}\})]$$

Phase I Meta-training of theta by minimising above equation

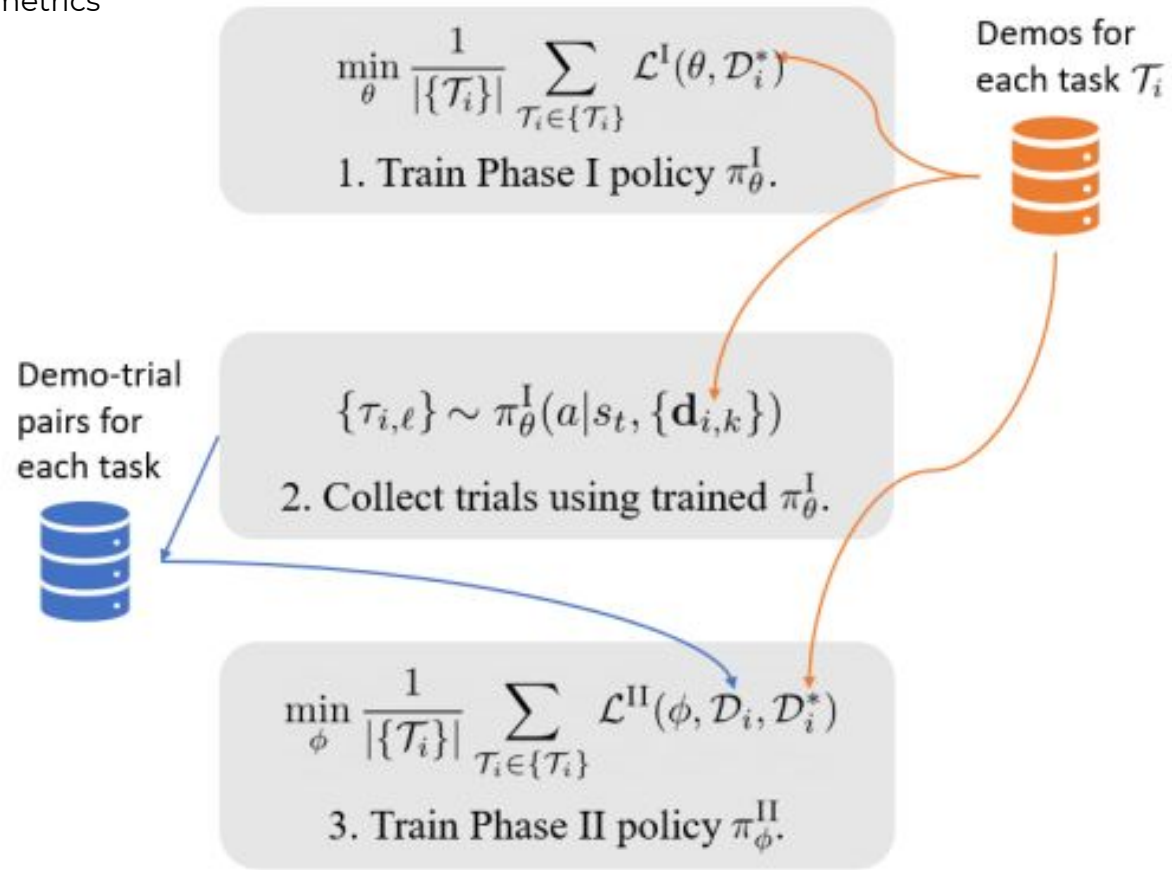
$$\min_{\theta} \frac{1}{|\{\mathcal{T}_i\}|} \sum_{\mathcal{T}_i \in \{\mathcal{T}_i\}} \mathcal{L}^I(\theta, \mathcal{D}_i^*)$$

Phase II loss for Task \mathcal{T}_i

$$\mathcal{L}^{\text{II}}(\phi, \mathcal{D}_i, \mathcal{D}_i^*) = \mathbb{E}_{(\{\mathbf{d}_{i,k}\}, \{\boldsymbol{\tau}_{i,\ell}\}) \sim \mathcal{D}_i} \mathbb{E}_{\mathbf{d}_i^{\text{test}} \sim \mathcal{D}_i^* \setminus \{\mathbf{d}_{i,k}\}} \mathbb{E}_{(s_t, a_t) \sim \mathbf{d}_i^{\text{test}}} [-\log \pi_{\phi}^{\text{II}}(a_t | s_t, \{\mathbf{d}_{i,k}\}, \{\boldsymbol{\tau}_{i,\ell}\})]$$

Phase II Meta-training of theta by minimising above equation

$$\min_{\phi} \frac{1}{|\{\mathcal{T}_i\}|} \sum_{\mathcal{T}_i \in \{\mathcal{T}_i\}} \mathcal{L}^{\text{II}}(\phi, \mathcal{D}_i, \mathcal{D}_i^*)$$



Implementation Algorithms

Algorithm 1 Watch-Try-Learn: Meta-training

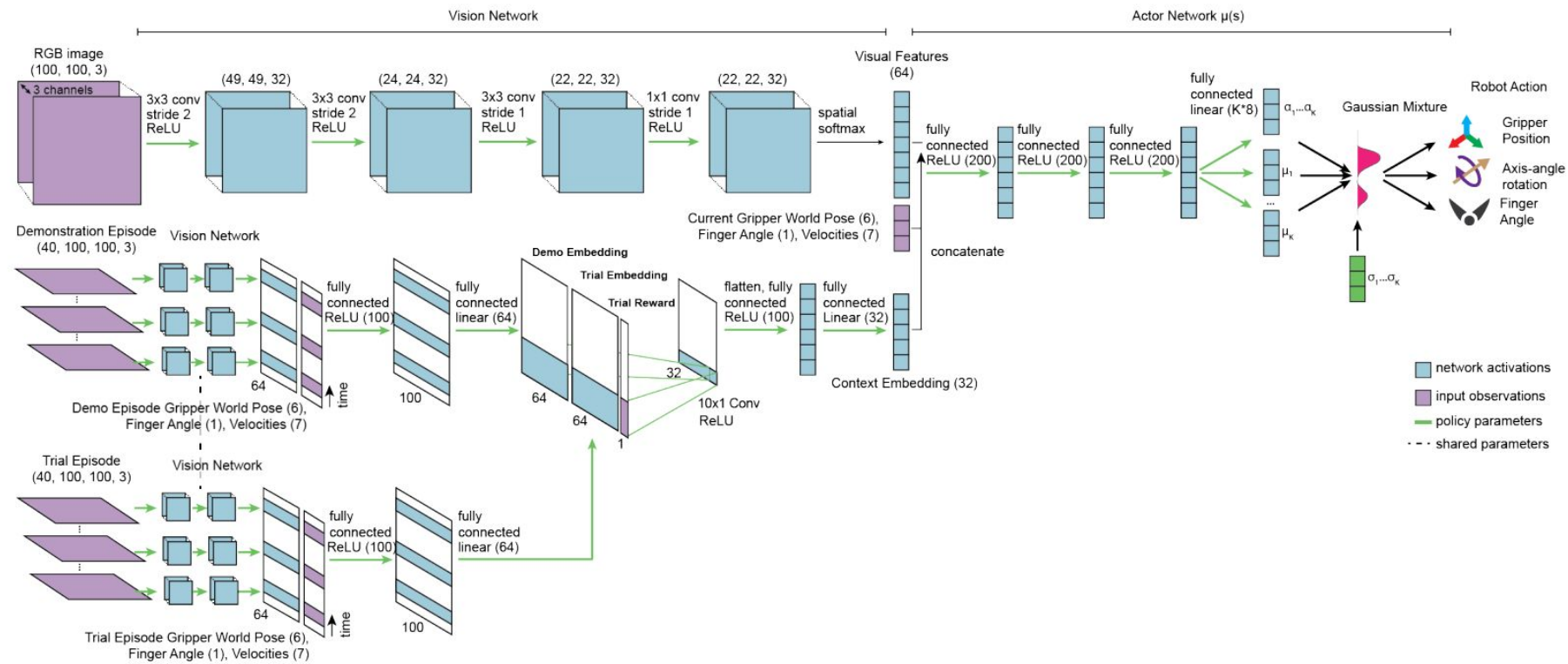
```
1: Input: Training tasks  $\{\mathcal{T}_i\}$ 
2: Input: Demo data  $\mathcal{D}_i^* = \{\mathbf{d}_i\}$  per task  $\mathcal{T}_i$ 
3: Input: Number of training steps  $N$ 
4: Randomly initialize  $\theta, \phi$ 
5: for step = 1,  $\dots$ ,  $N$  do
6:   Sample meta-training task  $\mathcal{T}_i$ 
7:   Update  $\theta$  with  $\nabla_{\theta} \mathcal{L}^I(\theta, \mathcal{D}_i^*)$  (see Eq. 1)
8: end for
9: for  $\mathcal{T}_i \in \{\mathcal{T}_i\}$  do
10:   Initialize empty  $\mathcal{D}_i$  for demo-trial pairs.
11:   while not done do
12:     Sample  $K$  demonstrations  $\{\mathbf{d}_{i,k}\} \sim \mathcal{D}_i^*$ 
13:     Collect  $L$  trials  $\{\tau_{i,l}\} \sim \pi_{\theta}^T(a|s, \{\mathbf{d}_{i,k}\})$ 
14:     Update  $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{(\{\mathbf{d}_{i,k}\}, \{\tau_{i,l}\})\}$ 
15:   end while
16: end for
17: for step = 1,  $\dots$ ,  $N$  do
18:   Sample meta-training task  $\mathcal{T}_i$ 
19:   Update  $\phi$  with  $\nabla_{\phi} \mathcal{L}^{II}(\phi, \mathcal{D}_i, \mathcal{D}_i^*)$  (see Eq. 3)
20: end for
21: return  $\theta, \phi$ 
```

Algorithm 2 Watch-Try-Learn: Meta-testing

```
1: Input: Test tasks  $\{\mathcal{T}_j\}$ 
2: Input: Demo data  $\mathcal{D}_j^*$  for task  $\mathcal{T}_j$ 
3: for  $\mathcal{T}_j \in \{\mathcal{T}_j\}$  do
4:   Sample  $K$  demonstrations
      $\{\mathbf{d}_{j,k}\} \sim \mathcal{D}_j^*$ 
5:   Collect  $L$  trials  $\{\tau_{j,l}\}$  with policy
      $\pi_{\theta}^I(a|s, \{\mathbf{d}_{j,k}\})$ 
6:   Perform task with re-trial policy
      $\pi_{\phi}^{II}(a|s, \{\mathbf{d}_{j,k}\}, \{\tau_{j,l}\})$ 
7: end for
```

Implementation

Model Description



Experiments



Similar Work

Inference from these prior implementations

They pre-trained a policy similar to BC, fine-tune for each meta-test, Average return of each method on held task using Soft Actor Critic.

- **Behavioral Cloning (BC)** (*Duan et al., 2016*)

A behavior cloning method that **does not condition on either demonstration or trial-and-error experience**, trained across all meta-training data. We train BC policies using maximum log-likelihood with expert demonstration actions.

- **Meta-Imitation Learning (MIL)** (*Finn et al., 2017b; James et al., 2018*)

A meta-imitation learning method that **conditions on demonstration data**, but does not leverage trial-and-error experience.

- **Behavioral Cloning with Soft Actor Critic** (*Haarnoja et al., 2018*)

We fine-tune a separate RL agent for each meta-test task

Results




Table shows that BC + SAC fine-tuning typically requires thousands of trial episodes per task to reach the same performance our meta-trained WTL method achieves after one demonstration and a single trial episode.

METHOD	SUCCESS RATE
BC	.09 ± .01
MIL	.30 ± .02
WTL, 1 TRIAL (OURS)	.42 ± .02
RL FINE-TUNING WITH SAC	
BC + SAC, 1500 TRIALS	.11 ± .07
BC + SAC, 2000 TRIALS	.29 ± .10
BC + SAC, 2500 TRIALS	.39 ± .11

Conclusion

- The Watch-Try-Learn (WTL) **approach enables** a natural way for non-expert **users to train agents to perform new tasks by demonstrating the task and observing and critiquing the performance of the agent on the task if it initially fails.**
- WTL achieves this through a unique combination of demonstrations and trials in the inner loop of a meta-learning system, where the demonstration guides the exploration process for subsequent trials, and **the use of trials allows the agent to learn new task objectives** which may not have been seen during meta-training.