1709.04905 - One-Shot Visual Imitation Learning via Meta-Learning

- **Yunqiu Xu**
- Other reference:
    - Presentation: https://www.youtube.com/watch?v=_9Ny2ghjwuY
    - Code: https://github.com/tianheyu927/mil

---

1. Introduction

- Challenge: learning each skill from a scratch is infeasible
- One-Shot Visual Imitation Learning via Meta-Learning
    - Reuse past experience to train the "base model", then adapt it to new task with only a single demonstration
    - Visual: use raw visual inputs
    - Meta-Learning: MAML C. Finn et.al. 2017
- Prior work : take task identity / demonstration as the input into a contextual policy
- Our work : learn parameterized policy, then adapt into new tasks through a few gradient updates

2. Related work

- In this work, the state of environment is unknown $\rightarrow$ we feed raw sensory inputs to learn it
- Two challenges for learning from demonstrations then applying it to real world:
    - Compounding errors: not in this work
    - **The need of a large number of demonstrations for each task**
- Why don't use Inverse RL:
    - How does it work : recover reward function from demonstrations

- Pros: reduce demonstrations, better than behavioral cloning
- Cons:
  - Requires additional robot experience to optimize the reward C. Finn et.al. 2016
  - Hard to evaluate learned reward, especially for high-dim data (image)
  - Gan-based IRL (e.g. GAIL) is hard to train
- How do we reduce the demonstrations: **share data across tasks**
  - First, use a dataset of demonstrations of many other tasks for meta learning, in this way we can build a base model
  - Then we can adapt this base model to new task with only a few demonstrations
  - Meta-learning is similar to transfer learning to some extent, the different is not the transfer on dataset, but the transfer on task
  - Take a simple instance, if the robot is learned to pick apple, orange, pear ... then it can pick peach easily

3. Meta-Imitation Learning

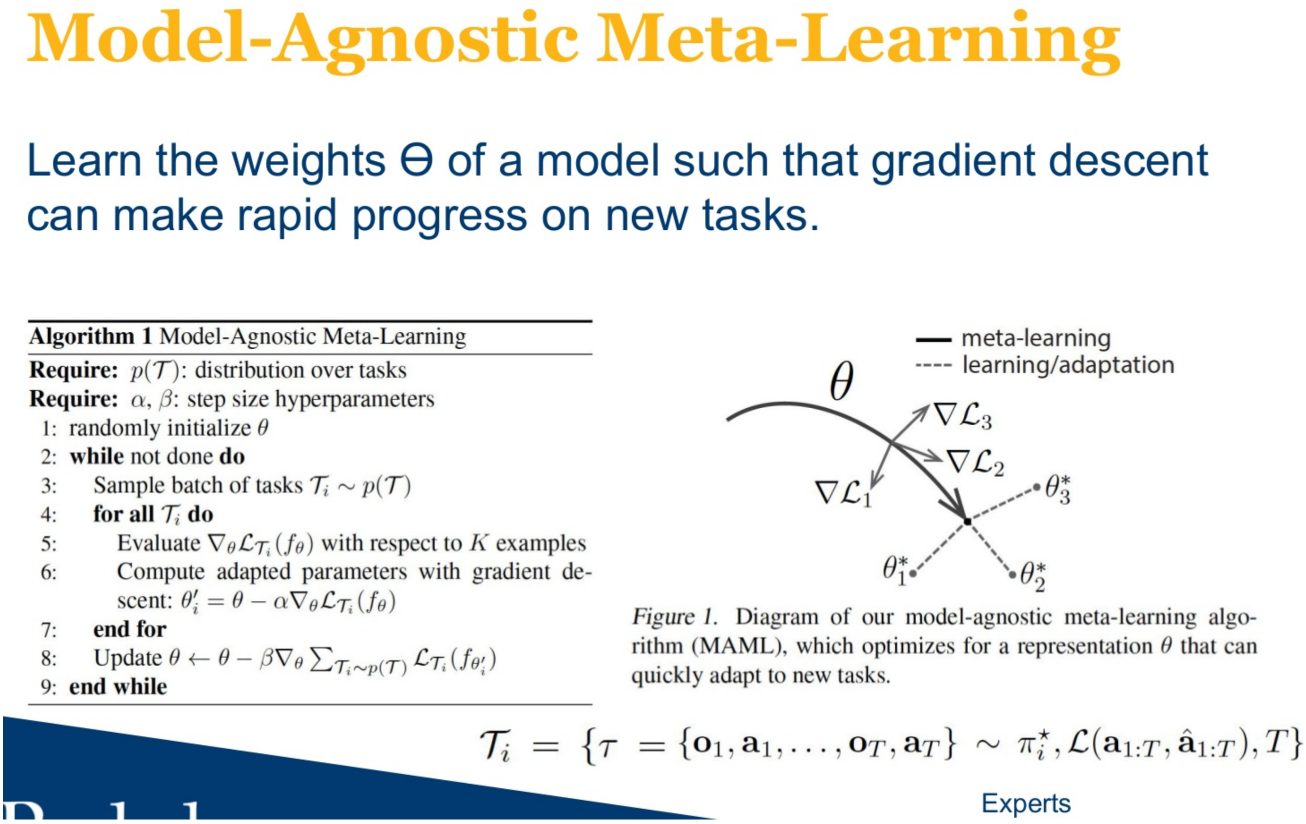
- Goal : learn a policy that can quickly adapt to new tasks from a single demonstration of that task
- Each imitation task $T_i = \left\{\tau = \{o_1, a_1, \dots, o_T, a_T\} \sim \pi_i^*, L(a_{1:T}, \hat{a}_{1:T}), T\right\}$
  - τ : a demonstration generated by policy π_i^*
  - $L(a_1, \dots, a_T, \hat{a}_1, \dots, \hat{a}_T) \to R$: loss function to give feedback
  - **Note that this form is different from original MAML**

3.1 MAML

- Consider a policy π with parameter vector θ
- Sample a task T_i from $p(T)$
- Train this task with K samples (adapt π to T_i to get new parameter θ')
- Test this task, then treat the testing error as the training error of meta-process (Use $\theta'_1, \dots, \theta'_n$, to update θ)
- Meta objective:

$$\min_{\theta} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta'_i}) = \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta - \alpha \nabla_\theta L_{T_i}(f_\theta)}) \qquad (1)$$

- Finally, you can adapt trained π to a new task with only a few data / gradient updates



3.2 Extend MAML to MIL

- o_t is the observation at time t, i.e. an image, while a_t is the action
- For demonstration trajectory τ, we use MSE to compute loss:

$$L_{T_i}(f_\phi) = \sum_{\tau_j \sim T_i} \sum_t ||f_\phi(o_t^{(j)}) - a_t^{(j)}||_2^2 \qquad (2)$$

- During meta-learning, we assume each task has at least 2 demonstrations, thus we can sample a set of tasks with two demonstrations per task
- Compute θ'_i with one demonstration $\rightarrow$ inner loop of meta-learning
- Use another demonstration to "test" θ'_i $\rightarrow$ update θ according to the gradient of meta-objective
- Meta-testing:
  - Sample a new task T and its one demonstration
  - This task can involve new goals or manipulating new, previously unseen

objects.
- Then we can adapt θ to this task

Algorithm 1 Meta-Imitation Learning with MAML

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample demonstration $\tau = \{\mathbf{o}_1, \mathbf{a}_1, ...\mathbf{o}_T, \mathbf{a}_T\}$ from $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using τ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2)
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample demonstration $\tau_i' = \{\mathbf{o}_1', \mathbf{a}_1', ...\mathbf{o}_T', \mathbf{a}_T'\}$ from $\mathcal{T}_i$ for the meta-update
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each τ_i' and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2
end while
return parameters θ that can be quickly adapted to new tasks through imitation.

3.3 Two Head Structure

- Why use this: more flexibility during adapting
- The parameters of pre-update head are not used for post-update head in final
- Modification : parameters of final layers are not shared, forming two heads
  - Change loss function as:

$$L_{T_i}(f_\phi) = \sum_{\tau_j \sim T_i} \sum_t ||W y_t^{(j)} + b - a_t^{(j)}||_2^2 \qquad (3)$$

    - $y_t^{(j)}$: post-synamptic activations of the last hidden layer
    - W, b : weights and bias for last layer
  - Then the meta-objective is about θ, W, b

$$\min_{\theta, W, b} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i'}) = \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta - \alpha \nabla_\theta L_{T_i}(f_\theta)}) \qquad (4)$$

3.4 Learning to Imitate without Expert Actions

- Why use this : it is more practical to simply provide a video of the task being performed
- We just simplify this problem by simplify the loss function as

$$L_{T_i}(f_\phi) = \sum_{\tau_j \sim T_i} \sum_t ||W y_t^{(j)} + b||_2^2 \qquad (3)$$

- **This can be a future question for more robust loss function**

4. Network Architecture

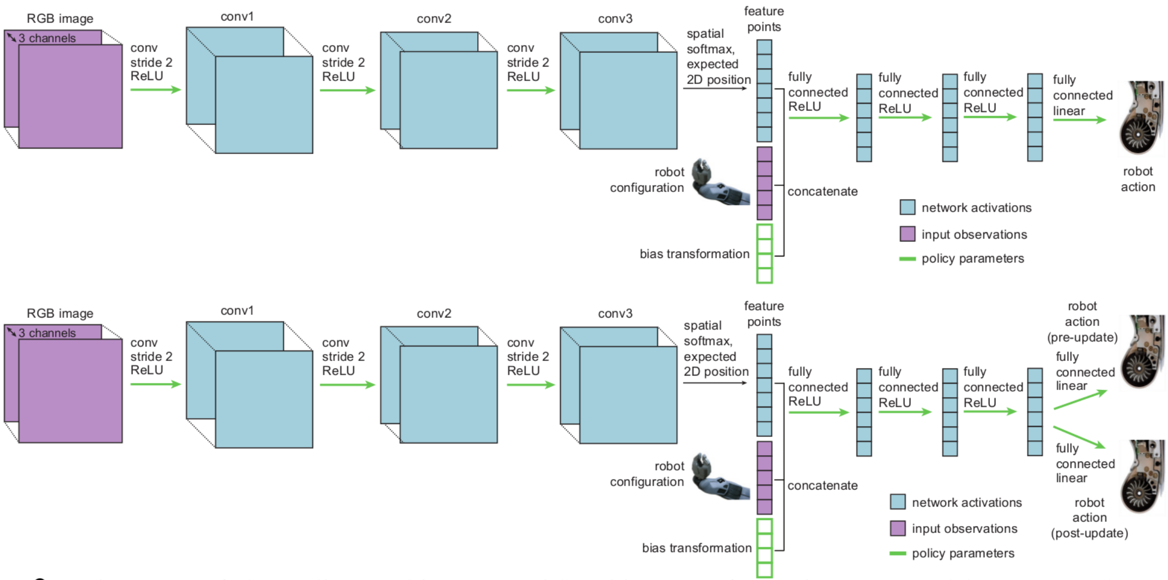


Figure 2: Diagrams of the policy architecture with a bias transformation (top and bottom) and two heads (bottom). The green arrows and boxes indicate weights that are part of the meta-learned policy parameters θ.

- **Layer normalization** after each layer
  - Data within a demonstration trajectory is highly correlated across time
  - Thus BN was not effective
- Bias transformation $\rightarrow$ improve the performance of meta-learning
  - Concatenate a vector of parameters to a hidden layer of post-synaptic activations
  - Thus vector is treated as same as other parameters during meta-learning and final testing

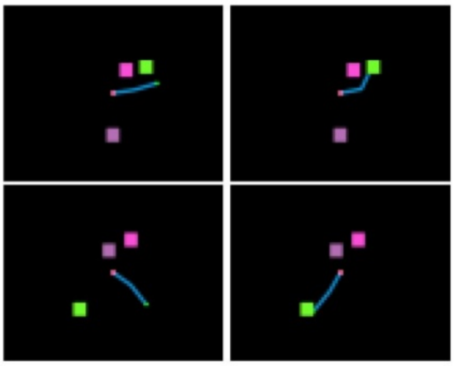
$$y = Wx + b \;\; \rightarrow \;\; y = W_1 x + W_2 z + b$$

    - z is the parameter vector form bias transformation
    - $W = [W_1, W_2]$
  - This modification **increases the representational power of the gradient**
  - Does not affect the representation power of the network itself

4. Experiment

- Questions:
    - Can a policy be learned to maps from image pixels to actions using a single demonstration of a task
    - How does our meta-imitation learning method compare to other one-shot imitation learning methods
    - Can we learn without expert actions
    - How well does our method scale to real world tasks
- Methods for comparison:
    - Our method
    - Random policy: output random actions from standard normal distribution
    - Contextual policy:
        - Input the final image of demonstration
        - Indicate goal and current image (observation)
        - Then output current action
    - LSTM:
        - Input demonstration and current observation
        - Output current action
    - LSTM + attention: only applicable to non-vision tasks
- Task 1 : simulated reaching
    - Goal: reach a target of a particular color
    - Both vision and non-vision versions are tested
    - meta-learning can handle raw-inputs
    - Our method can handle small dataset (demonstration) well
    - Bias transformation (bt) can perform more consistently across dataset sizes

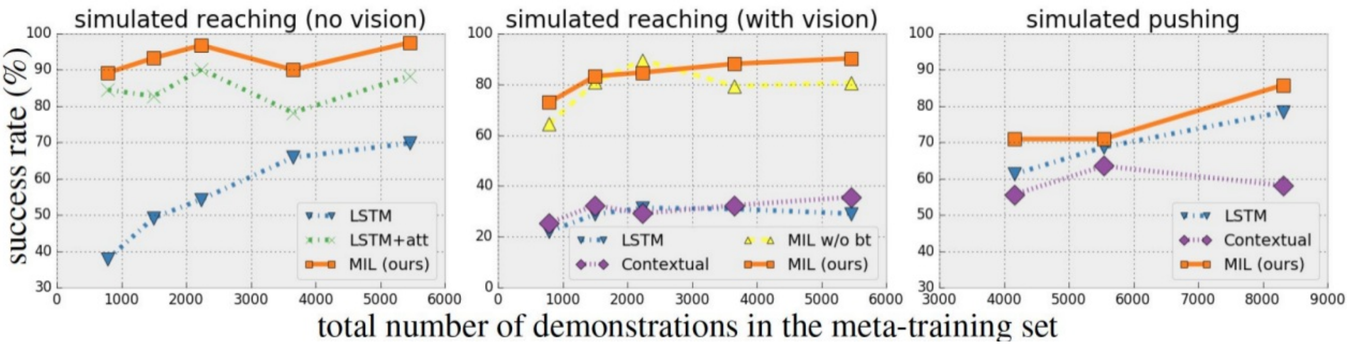
Simulated reaching



Task:

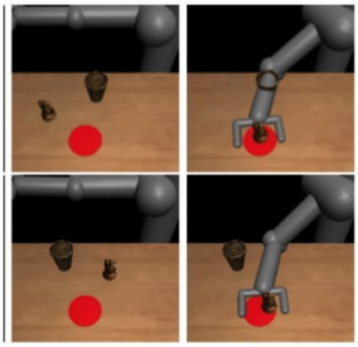
- reaching a target of a particular color
- Policy must learn to localize the target using the demonstration and generalize to new positions
- Meta-training must learn to handle different colors
- 150 tasks x 10 different trials per task

Simulated reaching



- Task 2 : simulated pushing

Simulated pushing

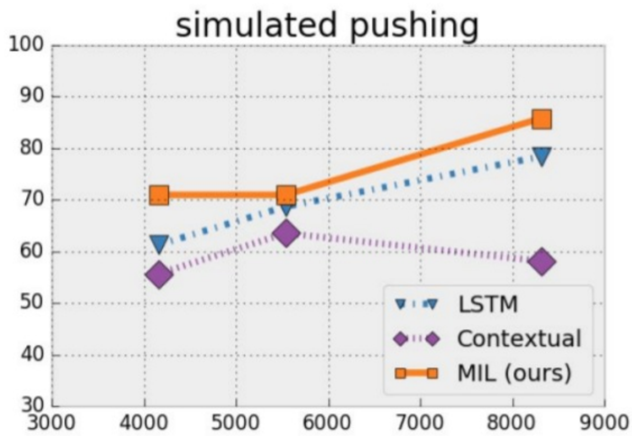


Task:

- ❏ A push is considered as success if the center of the target object lands on the red target circle for at least 10 timestamps.
- ❏ Each task is defined as pushing a particular objects
- ❏ 74 tasks x 6 different trials per task

method		video+state +action	video +state	video
LSTM	1-shot	78.38%	37.61%	34.23%
contextual	1-shot	n/a	58.11%	56.98%
MIL (ours)	1-shot	**85.81%**	**72.52%**	**66.44%**
LSTM	5-shot	83.11%	39.64%	31.98%
contextual	5-shot	n/a	64.64%	59.01%
MIL (ours)	5-shot	**88.75%**	**78.15%**	**70.50%**

Table 1: One-shot and 5-shot simulating pushing success rate with varying demonstration information provided at test-time. MIL can more successfully learn from a demonstration without actions and without robot state and actions than LSTM and contextual policies.



- Task 3 : real-world placing
  - Experiment goal : place a held item into a target container, such as a cup, plate, or bowl, while ignoring two distractors

Real-World Placing

subset of training objects | test objects | subset of training objects | test objects

Task:

Evaluate how well a real robot (PR2) can learn to interact with new unknown objects from a single visual demonstration.

Success: the held object landed in or on the target container after the gripper is opened

method	test performance
LSTM	25%
contextual	25%
MIL	**90%**
MIL, video only	**68.33%**

Table 2: One-shot success rate of placing a held item into the correct container, with a real PR2 robot, using 29 held-out test objects. Meta-training used a dataset with $\sim$100 objects. MIL, using video only receives the only video part of the demonstration and not the arm trajectory or actions.

5. Summary and Ongoing Work (On CoRL)

- Summary:
  - reuse prior experience when learning in new settings
  - learning-to-learn enables effective one-shot learning
- Ongoing work: one-shot imitation from human video → during demo, let human put the ball in cup