Watch, Try, Learn: Meta-Learning from Demonstrations and Rewards

Chen Zhao

ICLR 2020, https://arxiv.org/pdf/1906.03352.pdf

1 Motivation and Contributions

- Imitation learning enables autonomous agents to learn complex behaviors from demonstrations, which are often easy and intuitive for users to provide
- We develop a new meta-learning algorithm that incorporates elements of imitation learning with trial-and-error reinforcement learning
- We evaluate our algorithm and several prior methods on a challenging, vision-based control problem involving manipulation tasks from four distinct families of tasks: button-pressing, grasping, pushing, and pick and place
- We find that our approach can effectively learn tasks with new, held-out objects using one demonstration and a single trial, while significantly outperforming meta-imitation learning, meta-reinforcement learning, and prior methods that combine demonstrations and reward feedback
- BC + soft actor critic: In the gripper environment we study how much trial-and-error experience soft actor critic (SAC) (Haarnoja et al, 2018), a state of the art reinforcement learning algorithm, would require to solve a single task
- We proposed a meta-learning algorithm that allows an agent to quickly learn new behavior from a single demonstration followed by trial experience and associated rewards

2 Summary

- Imitation learning enables autonomous agents to learn complex behaviors from demonstrations, which are often easy and intuitive for users to provide.

- We find that our approach can effectively learn tasks with new, held-out objects using one demonstration and a single trial, while significantly outperforming meta-imitation learning, meta-reinforcement learning, and prior methods that combine demonstrations and reward feedback.
- Like prior few shot meta-learning works (Finn et al, 2017b; Duan et al, 2017), we hope to achieve few shot success on a task by explicitly meta-training our Phase I and Phase II policies to learn from very little demonstration and trial data.

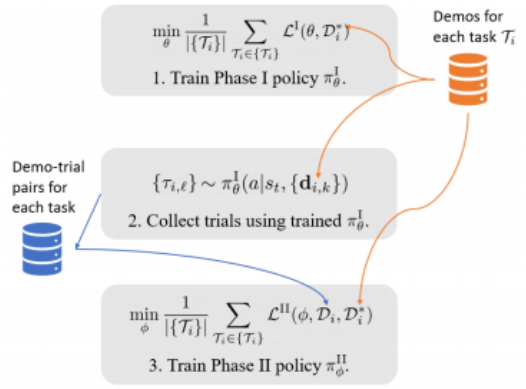

Figure 2: **Meta-training Overview**: First, we meta-train π_θ^{I} according to Eq. 2. Next, we collect L trial trajectories per meta-training task $\mathcal{T}_i$ in the environment using our trained π_θ^{I}. We denote the trial $\{\boldsymbol{\tau}_{i,\ell}\} \sim \pi_\theta^{\mathrm{I}}(a|s, \{\mathbf{d}_{i,k}\})$, and store the resulting demo-trial pairs $\{(\{\mathbf{d}_{i,k}\}, \{\boldsymbol{\tau}_{i,j}\})\}$ in a new dataset (blue). Finally, we meta-train π_ϕ^{II} according to Eq. 4.

- Prior meta-reinforcement learning work (Duan et al, 2016; Finn et al, 2017a) have addressed the issue of a changing trial distribution by re-collecting on-policy trial trajectories from the environment after every gradient step during meta-training, but this can be difficult in real-world problem settings with broad task distributions, where it is impractical to collect large amounts of on-policy experience.
- We aim to evaluate our method on challenging few-shot learning domains that span multiple task families, where the agent must use both demonstrations and trial-and-error to effectively infer a policy for the task.
- We evaluate how the following methods perform in those environments: BC: A behavior cloning method that does not condition on either demonstration or trial-and-error experience, trained across all meta-training data.
- 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.

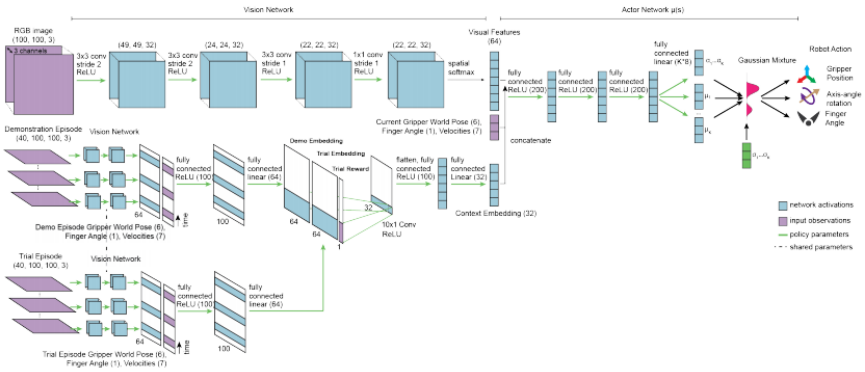

Figure 3: Our vision-based Phase II architecture: **Upper left**: we pass the RGB observation for each timestep through a 4-layer CNN with ReLU activations and layer normalization, followed by a spatial softmax layer that extracts 2D keypoints (Levine et al., 2016). We flatten the output keypoints and concatenate them with the current gripper pose, gripper velocity, and context embedding. **Upper Right**: We pass the resulting vector through the actor network, which predicts the parameters of a Gaussian mixture over the commanded end-effector position, axis-angle orientation, and finger angle. **Lower left**: To produce the context embedding, the embedding network applies a vision network to 40 ordered observations sampled randomly from the demo and trial trajectories. We concatenate the demo and trial outputs with the trial episode rewards along the embedding feature dimension, then apply a 10x1 convolution across the time dimension, flatten, and apply a MLP to produce the final context embedding. The Phase I policy architecture (Appendix Fig. 8) is the same as shown here, but omits the concatenation of trial embeddings and trial rewards.

- We pre-train a policy similar to BC, fine-tune for each meta-test Figure 4: Average return of each method on held task using SAC.
- Table 1 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.
- We proposed a meta-learning algorithm that allows an agent to quickly learn new behavior from a single demonstration followed by trial experience and associated rewards.
- 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.

Algorithm 1 Watch-Try-Learn: Meta-training

Input: Training tasks $\{\mathcal{T}_i\}$
Input: Demo data $\mathcal{D}_i^* = \{\mathbf{d}_i\}$ per task $\mathcal{T}_i$
Input: Number of training steps N
Randomly initialize θ, ϕ
for step $= 1, \cdots, N$ **do**
  Sample meta-training task $\mathcal{T}_i$
  Update θ with $\nabla_\theta \mathcal{L}^{\text{I}}(\theta, \mathcal{D}_i^*)$ (see Eq. 1)
end for
for $\mathcal{T}_i \in \{\mathcal{T}_i\}$ **do**
  Initialize empty $\mathcal{D}_i$ for demo-trial pairs.
  while not done **do**
    Sample K demonstrations $\{\mathbf{d}_{i,k}\} \sim \mathcal{D}_i^*$
    Collect L trials $\{\boldsymbol{\tau}_{i,l}\} \sim \pi_\theta^T(a|s, \{\mathbf{d}_{i,k}\})$
    Update $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{(\{\mathbf{d}_{i,k}\}, \{\boldsymbol{\tau}_{i,l}\})\}$
  end while
end for
for step $= 1, \cdots, N$ **do**
  Sample meta-training task $\mathcal{T}_i$
  Update ϕ with $\nabla_\phi \mathcal{L}^{\text{II}}(\phi, \mathcal{D}_i, \mathcal{D}_i^*)$ (see Eq. 3)
end for
return θ, ϕ

Algorithm 2 Watch-Try-Learn: Meta-testing

Input: Test tasks $\{\mathcal{T}_j\}$
Input: Demo data D_j^* for task $\mathcal{T}_j$
for $\mathcal{T}_j \in \{\mathcal{T}_j\}$ **do**
  Sample K demonstrations $\{\mathbf{d}_{j,k}\} \sim \mathcal{D}_j^*$
  Collect L trials $\{\boldsymbol{\tau}_{j,l}\}$ with policy $\pi_\theta^{\text{I}}(a|s, \{\mathbf{d}_{j,k}\})$
  Perform task with re-trial policy $\pi_\phi^{\text{II}}(a|s, \{\mathbf{d}_{j,k}\}, \{\boldsymbol{\tau}_{j,l}\})$
end for