Distributed Deep Multitask, and Transfer RL

Research Proposal

Ali Rahimi Kalahroudi*
Department of Math Science, Sharif University of Technology, Tehran, Iran

I. INTRODUCTION

The reinforcement learning community has made great strides in designing algorithms capable of exceeding human performance on specific tasks. These algorithms are mostly trained one task at the time, each new task requiring to train a brand new agent instance. This means the learning algorithm is general, but each solution is not; each agent can only solve the one task it was trained on. The learning process for each new agent instance requires a lot of time, and computational resources due to deep model architectures.

To tackle the stated issues, model compression, knowledge transferring, and multi-task learning techniques have been integrated into deep reinforcement learning. The approach that utilizes distillation technique to conduct knowledge transfer for multi-task reinforcement learning is referred to as *policy distillation* [7]. The goal is to train a single policy network that can be used for multiple tasks at the same time. In general, it can be considered as a transfer learning process with a student-teacher architecture. The knowledge is firstly learned in each single problem domain as *teacher* policies, and then it is transferred to a multi-task policy that is known as *student* policy.

Though some promising results have been shown recently, *policy distillation* for deep reinforcement learning suffers from the following challenges. First, the existing architectures involve multiple convolutional and fully-connected layers with a giant parameter size. This leads to a long training time for the models to converge. Second, to learn from multiple teacher policy networks, the student network needs to learn from a huge amount of data from each problem domain. Therefore, it is essential to develop an efficient algorithm, with efficient data collecting, and data sampling strategy to select meaningful data to update the network.

Recent advances in reinforcement learning have achieved significantly improved performance by leveraging distributed training architectures which separate learning from acting, collecting data from many actors running in parallel on separate environment instances [3, 5]. Distributed replay allows the *Ape-X agent* [5] to decouple learning from acting, with actors feeding experience into the distributed replay buffer and the learner receiving (randomized) training batches from it. In addition to distributed replay with prioritized sampling [8], *Ape-X* uses n-step return targets [9], the double Q-learning algorithm [10], the dueling DQN network architecture [11] and 4-framestacking. Ape-X achieved state-of-the-art performance on *Atari-57*, significantly out-performing the best single-actor algorithms.

II. OBJECTIVE

In this work, we aim to efficiently combine *policy distillation* and the *Ape-X* approaches, to speed up the process of knowledge transferring, and multi-task learning in *student-teacher* architecture. Reaching to a higher performance with our proposed algorithm in comparision with the state-of-the-art multi-task algorithms, and out-performing them on *Atari-57* (all available Atari games in Arcade Learning Environment [2]), and *DMLab-30* (a set of 30 tasks from the DeepMind Lab environment [1]) is our ultimate goal.

III. RELATED WORK

Transfer, and multi-task learning has been discussed in many recent works [3, 4, 6, 7, 12], and distributed learning algorithms are discussed in [3, 5].

IV. TECHNICAL OUTLINE

Consider the transfer learning objective which trains a student agent from teacher agents within a specific task such as the *Pong* game. The idea that combines both *Ape-X* algorithm, and *policy distillation* technique to make transferring the knowledge available can be articulated as :

- Define a *Learner* agent (like *Ape-X*) with a specific model architecture. The goal of this agent is to train itself using batches of data collected by some *Actors*, which will be defined later. The batch is like (s, p, q) where s is a specific state, p is action values taken under the softmax function generated by *Actor-Student* agent, and q has the same definition as p, but it is generated by *Actor-Teacher* agent. The *Learner* minimize the KL-divergence loss between p, and q values, as it is explained here [7].

* Also at: CafeBazaar Research Center, Tehran, Iran; ali.rahimikalahroudi@gmail.com

- Define each *Actor* agent to be made of two parts, an *Actor-Student*, and an *Actor-Teacher* agent. Every *Actor-Student* is an instance of the *Learner*, and every t steps updates its model parameters to be just like the *Learner* [5]. Every *Actor-Teacher* is an instance of the pre-trained agents on the task. This does not necessary mean that pre-trained agents (*teachers*) come from the same agent architecture; they could have different architectures, or even have been trained with different algorithms. The only common thing is their objective to do the same task (e.g. playing *Pong* game).

  Together *Actor-Student*, and *Actor-Agent* would generate, and collect data for the *Learner* to optimize its objective. The data is simply the (s, p, q) described at *Learner* section. As it is stated in *Ape-X* paper, this algorithm significantly relies on prioritized experience replay technique. To complete the *Actor* goal which is providing useful data for the *Learner*, every *Actor* would use simple prioritized sampling technique [8], or the *Hierarchical Prioritized Experience Replay* [12] for every of its (s, p, q) tuples and the rest of the algorithm is defined like *Ape-X*.

The above explanation combines the *Ape-X* algorithm with *policy distillation*, and hopefully could be used as a new algorithm for transfer learning in deep reinforcement learning.

Another idea to try is combining the *Ape-X*, with the above algorithm; meaning that the *Learner* would have two (possibly weighted) objective functions, first of which is come from *Ape-X* that is simple *DQN* objective, and the other is minimizing the KL-divergence loss between its own decisions, and its teachers.

In addition, multi-task learning could be achieved by adding teachers of different task domains, and applying the proposed algorithm, with (possibly) changes in *Learner* architecture.

---

[1] Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. Deepmind lab. *CoRR abs/1612.03801* (2016).

[2] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *CoRR abs/1207.4708* (2012).

[3] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *CoRR abs/1802.01561* (2018).

[4] Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task deep reinforcement learning with popart. *CoRR abs/1809.04474* (2018).

[5] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., and Silver, D. Distributed prioritized experience replay. *CoRR abs/1803.00933* (2018).

[6] Parisotto, E., Ba, L. J., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR abs/1511.06342* (2015).

[7] Rusu, A. A., Colmenarejo, S. G., Gülçehre, Ç., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation. *CoRR abs/1511.06295* (2015).

[8] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *CoRR abs/1511.05952* (2015).

[9] Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning 3* (1988), 9–44.

[10] van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *CoRR abs/1509.06461* (2015).

[11] Wang, Z., de Freitas, N., and Lanctot, M. Dueling network architectures for deep reinforcement learning. *CoRR abs/1511.06581* (2015).

[12] Yin, H., and Pan, S. J. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *AAAI* (2017).