CS7610 Final Project Proposal

David Klee and Luke Roberto

1 Introduction

For our final project we seek to build a decentralized training system for a reinforcement learning agent. Current methods using centralized servers require expensive synchronization efforts of all workers to ensure consistency in updates to the global model. In order to combat this, we want to explore the methods used in Macua et. al [1] to train a global policy in a decentralized manner.

2 Background

There are a few major issues in Deep Reinforcement Learning (DRL):

1. Sample efficiency
2. Exploration
3. Use of memory-inefficient replay buffers

Many modern methods of scaling up DRL for production use a parameter server-style system as detailed by Li et al. [2] and shown in fig 1. The approach is to essentially create several ”workers” that will run some policy on an instance of the agents environment. All of these workers will act in parallel instances of the environment and share experiences with one another (eliminating the need for replay buffers).

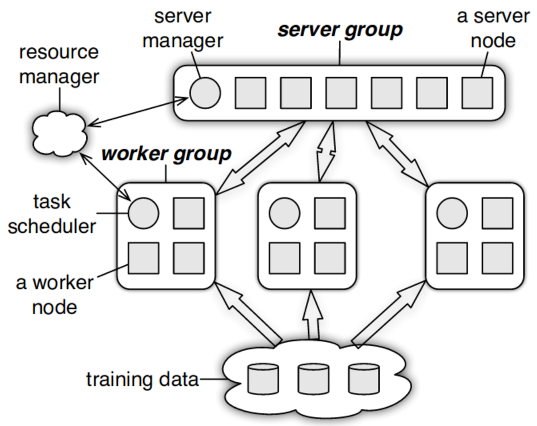

Figure 1: Abstract Implementation of Parameter Server

The manages the sample inefficiency because we have parallelized the number of agents that can collect experience, and also helps with exploration because each worker can run a different type of exploration policy to create more diversity. The workers will periodically synchronize and communicate the gradient updates to a main parameter server. This server is usually implemented as some fault-tolerant Paxos system (or other state replication protocol) to ensure that the gradient updates are consistent and non-blocking.

While very successful, and widely implemented in several well known projects, there are a few drawbacks to this system architecture. One major drawback is the added complexity moving from serial algorithms to these fault-tolerant distributed system designs. Infrastructure needs to built that organizes data flow between workers, and then from workers to parameter servers. Depending on the specific reinforcement learning algorithm used, the use of the parameter will also introduce blocking in the synchronization step between worker nodes. This can slow the total throughput of these systems down, and even if this blocking is removed then the sample efficiency of such approaches is even worse.

RL does not require that updates to the model be ordered or even that synchronization of the worker nodes needs to happen. Macua et al. [1] propose reformulating the training problem as a multi-agent reinforcement learning problem. Instead of solving the problem with complex system architectures, the proposal is to decentralize the learning of the global model parameters, as shown in figure 2.

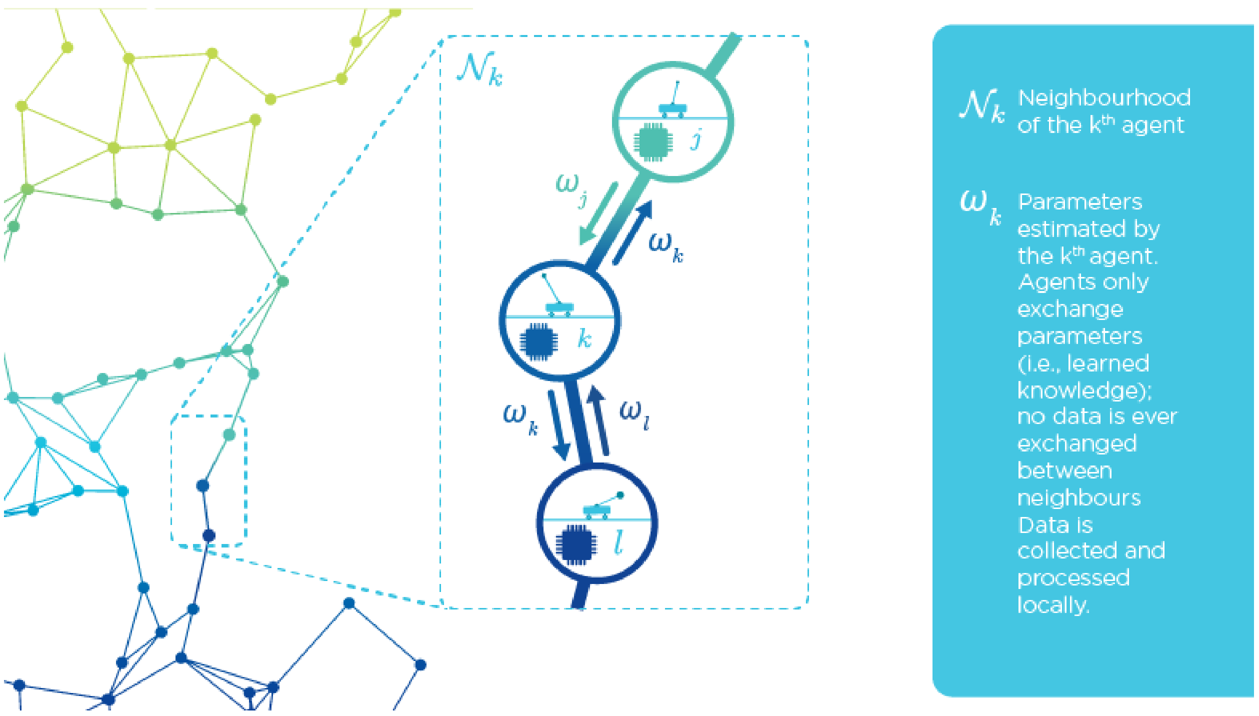
$\mathcal{N}_k$
j
ω_j
ω_k
k
ω_k
ω_l
l
$\mathcal{N}_k$ Neighbourhood of the kth agent
ω_k Parameters estimated by the kth agent. Agents only exchange parameters (i.e., learned knowledge); no data is ever exchanged between neighbours Data is collected and processed locally.

Figure 2: Abstract implementation of Decentralized Training

The main idea of this system is to train each agent without a replay buffer and only have them communicate with their neighbors. This solves all the major issues listed above still because we can parallelize experiences to tackle sample efficiency and increase diversity in exploration.

3 Minimum Viable Project

The minimum viable project for us to present at the end of the semester is an implementation of the algorithm in the paper. We will also test it in a variety of ways to see its robustness to certain types of faults. We will test network partitions, the diffusion time to get an optimal global policy, and also Fail-Stop faults. We will also look into bench-marking against David Slayback's implementation of distributed actor-critic as well and see the performance difference between our method and a centralized one. If time permits, we will use the information gleaned from testing to either recommend or implement solutions to improve performance.

References

[1] Daniel Garc. Diff-DAC: Distributed Actor-Critic for Average Multitask Deep Reinforcement Learning. 2017.

[2] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-yiing Su. Scaling Distributed Machine Learning with the Parameter Server.