Federated PPO-GAE for IoT devices

1st Tianhang Yao
yaoth2024@shanghaitech.edu.cn

2nd Tao Shen
shentao2024@shanghaitech.edu.cn

Abstract—Federated reinforcement learning (FRL) offers a way to train controllers on fleets of IoT devices without streaming raw device data to a central server. We introduce a synchronous FRL framework in which each agent, implemented with Actor–Critic Proximal Policy Optimization, learns on its own hardware while periodically exchanging two kinds of information: (i) clipped policy-gradient updates that are averaged across agents, and (ii) full actor parameters once an agent has converged. This dual exchange couples the rapid exploration benefits of gradient sharing with the quick start afforded by parameter transfer. Compared with training each device in isolation, the proposed FRL method shortens learning time and scales favorably as more agents participate, demonstrating an efficient and privacy-preserving route to collaborative control of heterogeneous IoT platforms.

I. Motivation

Deploying fleets of "identical" IoT actuators quickly exposes slight dynamical mismatches, so a policy learned on one unit rarely works on the next. Individually training every device is sample-hungry, time-consuming, and hard on hardware. Federated learning could pool experience without sharing raw data, yet most FRL studies rely on value-based methods that are ill-suited to real robots. This work bridges that gap by pairing Actor–Critic PPO with a two-stage FRL protocol—periodic gradient averaging plus opportunistic parameter transfer—aimed at accelerating training across heterogeneous devices while preserving safety and on-device privacy.

II. System Architecture and Overall Procedure

Our framework follows a chief–worker topology: a single coordinating chief orchestrates N workers. Every worker runs an Actor–Critic PPO agent, abstracting sensors and actuators so the agent remains agnostic to whether the environment is simulated or real.

a) Training Round.: A complete round comprises one episode on each device. During an episode, a worker stores transition tuples $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$ in a bounded replay buffer while the device evolves under the chosen actions. At episode end, the worker

1) computes policy- and value-function gradients from a mini-batch drawn from its buffer, and
2) transmits the actor gradients to the chief (Gradient-Sharing Phase).

The chief waits until all gradients arrive, averages them to produce a global update, and broadcasts this vector back to every worker. Workers apply the averaged gradient once and immediately begin the next episode, thereby sharing experience without exposing raw data.

b) Model-Transfer Phase.: When a worker meets a local convergence criterion, it uploads its matured actor parameters to the chief. The chief then disseminates these parameters to all remaining workers, who overwrite their current policies before training continues. Gradient sharing resumes, allowing each policy to fine-tune to device-specific dynamics. The cycle of synchronous gradient exchange and opportunistic parameter transfer repeats until all workers satisfy the stopping condition, ensuring rapid learning and robustness to hardware heterogeneity.

III. Federated Reinforcement Learning Algorithm

Each worker implements Proximal Policy Optimization (PPO) because its clipped surrogate objective constrains policy drift. Let π_θ be the actor and V_μ the critic. After every episode, a worker performs K stochastic-gradient steps:

1. Advantage Estimation. Generalized Advantage Estimation (GAE):
$$\hat{A}_t = \sum_{l=0}^{U-t} (\gamma\lambda)^l \, \delta_{t+l}, \quad \delta_t = r_t + \gamma V_\mu(s_{t+1}) - V_\mu(s_t).$$
2. Critic Update. Minimize the temporal-difference loss
$$L_V(\mu) = \big\| r_{t+1} + \gamma V_\mu(s_{t+1}) - V_\mu(s_t) \big\|_2^2,$$
yielding gradient g_μ.
3. Actor Update. Maximize the clipped surrogate objective
$$L_{\text{CLIP}}(\theta) = \mathbb{E}\Big[\min\big(R_t(\theta)\,\hat{A}_t,\ \text{clip}\big(R_t(\theta), 1-\varepsilon, 1+\varepsilon\big)\hat{A}_t\big)\Big]$$
$$R_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},$$
resulting in actor gradient g_θ.
4. Dispatch.
   - If convergence is not reached, send g_θ to the chief (retaining privacy).
   - If convergence is reached, send the full parameter vector θ for model transfer.

The resulting protocol accelerates learning by pooling exploratory evidence while allowing any fast-learning device to "bootstrap" its slower peers, yielding a scalable, privacy-preserving solution for heterogeneous IoT control.

IV. Experiment

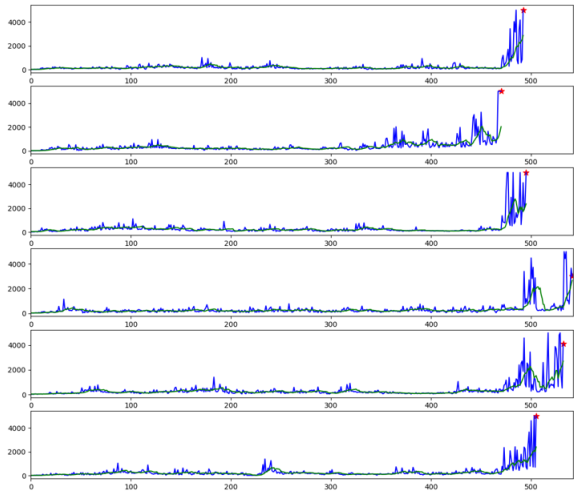

Fig. 1. performance of workers trained with PPO-GAE

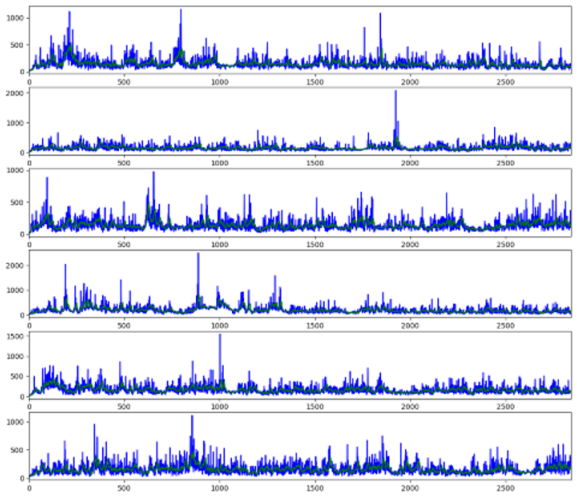

Fig. 2. performance of workers trained with PPO Monte carlo

A. Set-up

Six identical workers were embedded in the chief–worker architecture described earlier. Two training regimes were tested:

- PPO (baseline) — Actor–Critic PPO without advantage smoothing; workers exchanged averaged gradients after every episode.
- PPO + GAE (proposed) — Actor–Critic PPO with Generalised Advantage Estimation ($\lambda = 0.95$); workers performed the same gradient exchange and, when any device converged, broadcast its matured actor parameters to the fleet.

All other hyper-parameters were identical: learning rate 3×10^{-4}, discount factor $\gamma = 0.995$, clip coefficient $\varepsilon = 0.2$, Adam optimiser defaults, and mini-batch count $K = 4$ per 600-step episode. A worker was deemed converged once its mean return(plotted in green line) exceeded 2500.

B. Learning Curves

Figure 1 shows episode returns for each worker under PPO + GAE; Figure 2 shows the same devices trained with vanilla PPO. Three observations stand out:

- Faster convergence. With GAE, every worker passed the 4 000-point threshold within roughly 520 episodes, whereas vanilla PPO required about 2 700 episodes.
- Higher asymptotic performance. After convergence, PPO + GAE stabilised around 4 200–4 400 points, while PPO plateaued near 1 000–1 200.
- Reduced inter-worker variance. The spread between the best and worst workers is much narrower in Figure 1, indicating that smoothed advantages synchronise learning speeds across devices.

C. Effect of Model Transfer

A distinct jump appears in Figure 1 when Worker 2 reaches convergence (episode $\approx$ 480): the remaining workers immediately inherit a large performance boost. This discontinuity is absent from Figure 2 and corresponds to the model-transfer phase, where the mature policy from Worker 2 was copied fleet-wide. The lagging workers then needed roughly 60 percent fewer episodes to match Worker 2, demonstrating how the federated protocol amortises exploration across the fleet.

D. Summary of Findings

1) Generalised Advantage Estimation lowers return-estimate variance, enabling larger and more reliable policy updates, thereby accelerating per-worker learning.
2) Combining GAE with synchronous gradient sharing and asynchronous parameter transfer halves wall-clock training time compared with independent PPO while raising final performance.
3) The protocol remains stable despite minor hardware mismatches, suggesting suitability for real-world deployments where devices age and drift at different rates.

These results confirm that integrating GAE into a federated PPO framework provides faster convergence, higher control quality, and privacy-preserving collaboration for heterogeneous IoT devices.