To all reviewers: Thank you for your comments and suggestions. We would like to reemphasize that the focus of this paper is on computational efficiency in Deep-RL training, and we believe that GALA is a promising approach to accomplish this. In particular, we would like to point out that GALA-A2C maintains the sample efficiency of synchronous A2C (Fig 1 b–c), while exhibiting consistently comparable or superior computational efficiency across environments (Fig 1 d–g, and Fig 3).

Reviewer 1: In the experiments in Figs 1b–f, both A2C and GALA-A2C use 64 simulators. In A2C, workers exactly average their gradients using ALLREDUCE, so 4 workers with 16 simulators each is equivalent to a single A2C worker with 64 simulators. GALA-A2C uses gossip for approximate averaging, so this equivalence does not hold. Fig 1b shows the reward as a function of total number of steps taken across all simulators. GALA-A2C executes steps faster because workers run asynchronously, whereas in synchronous A2C, all simulators must be synchronized before each update.

Using many simulators leads to highly correlated observations (and gradients), which destabilizes training. You are correct in that sparsifying P helps decorrelate observations at different agents to some extent, but when they become too decorrelated (high disagreement) averaging their gradients may slow down learning. P is inherently stochastic in GALA due to asynchronous (non-deterministic) execution. Exploring effects of varying/controlling P further is an interesting line of future work.

Reviewer 2: To the best of our knowledge, the results in Props. 1 and 2 are novel. Related results are available in the literature for averaging or for optimization, but the setting here is more general (e.g., no assumptions about gradient smoothness) and we bound disagreement whereas results in gossip-based optimization typically bound suboptimality.

Regarding the benefits of noise, Fig 2 shows that when agents use gossip for approximate aggregation, their parameters are not identical and their gradients become less correlated. Fig 1a provides some evidence that GALA-A2C is more stable than A2C. We agree that the evidence could be strengthened, and we will soften the claim in lines 58–60.

L6 of Algo 1 checks the receive buffer (non-blocking) and only executes L7 if a message has been received. We will clarify that the "Broadcast" in L5 is implemented using asynchronous non-blocking sends/copies.

We agree that "convergent" may be confusing and will switch to "successful" in the revision. The nominal score is with respect to standard 16-simulator A2C scores.

The caption of Table C.1 in Espeholt et al. [2018] mentions 200M environment steps, and Table G.1 in that paper lists "Action repetitions" as 4, which we collectively interpret to mean 200M steps, not frames. We have contacted the authors of that paper for confirmation but haven't received a response yet. In our paper, Fig 1 uses actions sampled from the policy whereas Table 1 uses the argmax policy. We include Table 1 to be able to compare with results they report in the paper (from Table C.1).

In fact, the results in Fig 1a are statistically significant. A paired t-test returns a p-value of 0.0001303, indicating a $> 99.98\%$ chance that the GALA-A2C distribution of points in Fig 1a have a larger mean than the A2C distribution. Recall that each point in the figure corresponds to 10 runs, and we observe the same trend across an exponential sweep (2^6–2^{10} simulators) in all 6 games. This corresponds to 600 independent runs.

We did not use random start action mitigation in the reported experiments. Thank you for your additional suggestions, we will address these in the revised version of the paper.

Reviewer 3: The hyperparameters we use for all baselines are the latest published in the literature that we are aware of for each respective algorithm: Stooke & Abbeel [2018] for A2C and A3C, and Espeholt et al. [2018] for IMPALA. We chose to do an in-depth comparison on 6 specific games, reporting all runs from 10 independent seeds per algorithm per game, rather than spreading experiments across more games, to get higher statistical confidence. Those six specific games are the same ones used in Stooke & Abbeel [2018] (and include all five games used in Mnih et al. [2016]) which also focuses on computational efficiency and serves as the primary baseline.

Multiple agents in A2C is exactly as you describe, different shards on different devices whose gradients are summed using ALLREDUCE as in Stooke & Abbeel [2018]. The A2C batch size is proportional to the number of simulators, and when increasing the number of simulators we adjust the learning rate following the recommendation in Stooke & Abbeel [2018]. Fig 4 shows the average performance over all 10 runs, with 95% confidence intervals shaded when using A3C with 64 simulators. The A3C results in Mnih et al. [2016] show best 5 of 50 runs using 16 simulators, hence the difference. The implementation of Mnih et al. [2016] only leverages CPUs. To provide a more fair comparison to A3C in terms of computational efficiency (which is the focus of this work), we compare with the more competitive GPU-based implementation of A3C from Stooke & Abbeel [2018]. We will add results for CPU-A3C with 16 simulators in the appendix.

Thank you for your other suggestions. We will add other baseline algorithms to Fig. 3 in the revision. We also agree that a comparison/integration with methods like R2D2 is an interesting direction for future work.