We would like to thank the reviewers for their time. We want to reiterate our primary objective is to **make PBT more efficient**, such that it can be **effective with a small computational budget**. This was motivated by our own frustration with brittle RL hyperparameters, without the budget to run PBT. It is great to see all reviewers got this: R1 and R2 both saw the benefits in RL, saying this "might be of particularly importance for reinforcement learning algorithms", and "empirical results shown in the RL setting are very good" respectively. Meanwhile R2 noted the improvement in particular vs. PBT, and R4 appreciated both the theoretical results and variety of experiments. We now seek to clarify all concerns, and hope this is sufficient to consider raising scores.

BO Comparison R1 R2 We tested vanilla BO (sequential, EI, RatQuad) on the BipedalWalker task. We used 500k samples for each trial, to have sufficient number of evaluations in 4M and 8M timesteps (to compare vs. $B \in \{4, 8\}$). In Fig 1 we see BO performs poorly with the 4M budget, and for the 8M budget still underperforms PB2. This sequential method should be superior in performance to batch BO methods, since each trial is completed with full knowledge of all previous trials. The improvement therefore represents gains from updating hyperparameters on the fly (also see Fig 5 d) of Jaderberg et al. 2017). We also tried to compare against BOHB, however the ray tune version does not work for RL. We do not think it is fair to compare codebases for RL (see: Engstrom, ICLR 2020), so will seek to resolve this in time for the CRC. In the ASHA paper they outperform BOHB, so we feel this is a strong non-PBT baseline.

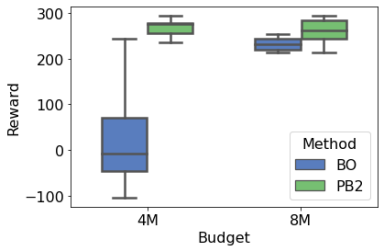

Figure 1: BipedalWalker, best rewards.

Larger Population Sizes R2 R4 We agree it is important to assess PB2 with $B > 8$. To answer this, we ran the BipedalWalker experiment with $B = 16$. As we see in Figure 2, both PBT and PB2 achieve optimal rewards (> 300), but PB2 is still more efficient. We hope this provides confidence in our method's effectiveness with more resources.

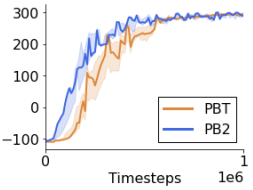

Figure 2: BipedalWalker, median, 5 seeds, $B = 16$.

Larger Supervised Experiments R2 R3 We included the SL experiment to show our method generalizes beyond RL, as noted by R4. The relative performance of the hyperparameter algorithms should be agnostic to the architecture, and PB2 outperforms other competitive baselines **across five seeds**. We used a medium sized CNN, as this was the most powerful model we could realistically train with our compute. Larger networks are prohibitively expensive for our lab. Instead, we allocated our resources on larger RL settings such as Atari (which are more CPU intensive). For reference, a similarly accurate CNN model was used for the first set of experiments in ASHA (as a POC). The main result in BOHB was a CNN on CIFAR-10, but they used 19 workers, each with 2 GPUs. However, their RL experiments were toy. In light of this, we think expecting SOTA SL experiments on a paper focused on RL is unreasonable.

What is $fg_{t+1}(x)$? R1 R2 It should be written as $g_{t+1}(x)$, without f. Thank you for catching this.

Next we address individual comments in more detail.

R1: **Wall-clock comparison** For RL, the GP-bandit step takes a trivial amount of time compared to querying a simulator or computing gradients, given we only have < 10 hyperparameters we have a tiny dataset. If we have vast computational resources and we wish to optimize $\gg 10$ hyperparameters for $B > 20$, it may be required to use more scalable GP methods. However, *this is not the problem we are trying to solve*. **PB2 only slightly outperforming ASHA** The main purpose of this paper is to improve PBT, and our main goal is to show this. In addition, we also show PB2 outperforms state of the art (ASHA). **Sec 3.1** To be precise, if (1) $\omega = 0$ and (2) the number of parallel agent $B = 1$, then our model reduces to GP-UCB. The reduction factor is 3, this is the default in the ray tune implementation.

R2: Thank you very much for your comments, we are glad you feel this is an improvement over PBT, in particular making it more usable, which was always our goal. We dont assume the convexity of the function. Instead, we make a mild condition on the compact and convex input space so that the input space is continuous and not segmented, e.g., the case of non-convexity. This mild assumption is popular in GP bandit literature (see Srinivas et al, 2010). We always have $B \ll T$ because the number of update step T typically goes beyond thousands to millions while the batch size $B = 4, 8, 16...$ is much smaller and limited to the parallel facility we have. **Hyperparameter Schedules** We disagree the hyperparameters are random, for the learning rate in particular, you can see the GP balances exploration and exploitation by either selecting points from a single mode or right at the boundaries.

R3: **Distinguishing features of our proof** Our proof makes two significant extensions beyond TV-GP-UCB (Bugonovic et al, 2016). First, we improve the bound from C.2 from $\tilde{N}^3$ to $\tilde{N}^{2.5}$ (smaller is tighter and better). Second, we extend the regret bound to the batch setting including the new Lemma 8,9,10 (in the Appendix). **Section 5.3** The hyperparameters and ranges are shown in Table 9, in the Appendix (top of p14). **Larger datasets using larger networks** See above: we hope this is not the primary reason for rejecting our work. **Optimal network architectures**, given recent works using BO for NAS we think this is a logical next step, and something we are excited about. NAS for RL is a nascent field.