We would like to thank the reviewers for their comments. Reading the reviews, we got the impression that the reviewers enjoyed certain parts in our paper but had concerns about other parts. We hope that our rebuttal addresses these concerns and would like to encourage the reviewers to consider changing their scores if this is the case.

Novelty. RL research suffers fundamentally from a major practical research bottleneck - the ability to set hyper-parameters to reasonable values. As RL researchers explore more complex architectures with more pieces, they introduce more hyperparameters. Tuning them often becomes a practical barrier to find signals of improvement over SOTA, perhaps limiting such research to groups with large computational budgets. Practically speaking therefore, the efficient auto-tuning of hyperparameters may be among the biggest step changes we can make to our algorithms. If the reviewers believe that self-tuning has the potential to be transformative to the practice of RL research, progress in that direction is worthy of publication.

The most significant contribution of this paper is a demonstration that self-tuning leads to big gains in performance for both STAC and STACX. We believe that achieving SOTA results in established and challenging benchmarks is **one** important way to make progress in ML research. It is easier to improve an under performing agent in a small environment than to improve a SOTA algorithm on an established benchmark. Our paper demonstrates significant empirical gains from using meta gradients in all benchmarks – the most significant gains that had been reported from using meta gradients so far. We believe that these findings would be of interest to the NeurIPS community.

We are aware that only pushing for SOTA results may lead to over fitting to certain domains and result in shallow understanding. To address these issues, we performed two types of experiments. 1. We demonstrated that the self-tuning ideas transfer from the ALE domain to control environments; we measured relative improvement from self-tuning in all the benchmarks. 2. We performed extensive ablative studies, robustness studies and visualizations of adaptivity. These experiments suggest self-tuning as many differentiable hyperparameters as possible, that self-tuning is quite robust, and that it can even discover theoretically sensible properties (contraction in V-trace).

Auxiliary tasks and self tuning (R3 & R4). R4 suggested that these are orthogonal ideas, and R3 had questions regarding the ablative study. Auxiliary tasks are a good example of the above-mentioned increasing complexity of RL architectures that introduces many new hyperparameters but end up being beneficial by improving the representation power of DRL agents. This is where self-tuning comes into the picture. In Section 3 we explain that by using self-tuning, we can instead introduce auxiliary tasks while only adding a single new hyper parameter (the number of tasks). We show that *without* self-tuning, these auxiliary tasks achieve similar performance to other auxiliary tasks (246, similar to the unreal agent with 252). But, with self-tuning, there is only one hyperparameter and we achieve much higher performance. The ablative analysis in Figure 3 (b) shows what happens when we tune different subsets of the meta parameters in with auxiliary tasks (STACX, blue bars). Specifically, to answer Q2 by R3, self-tuning all the meta-parameters compared to only the discounts resulted in improving the median from 262 to 364.

Leaky V-trace (R3 & R4). R3 mentioned that the leaky V-trace solution seems a bit adhoc and R4 was not sure why we chose to use it over other off policy evaluation mechanism. Leaky V-trace was a contribution to making the trade-offs among bias, variance and contraction in a **differentiable** (and thus self-tunable) and interpretable form as we discuss in Section 3 and as quantified in the proof. We demonstrated that in most of the games, the Leaky V-trace parameter self-tunes from V-trace to importance sampling as training progresses and the policy changes less rapidly (the adaptivity curves). In addition, the ablative analysis confirms that trying to differentiate the thresholds directly does not perform well. More broadly, many theoretically grounded loss functions often satisfy different trade-offs and self-tuning can help to mix and adapt them. Leaky V-trace provides a simple but important evidence that its possible to do that.

R3. We did not address non differential hyper parameters in the paper, but a recent paper "Online Hyper-parameter Tuning in Off-policy Learning via Evolutionary Strategies" by Tang et al does.

R4. **Random seeds.** As a design principle we believe that given a fixed budget of compute it is more meaningful to test an algorithm across a wider range of environments than across a wider range of random seeds. The reasoning behind this is that averaging across environments has more variability than averaging across seeds, but still averages across the randomness of the algorithm. In particular in the ALE benchmark, variability across seeds is minimal, thus 3 seeds of 57 environments is a reasonable compromise that has been adapted by the community. **Prior work.** We did cite a few works on hyper parameter tuning and specifically for adapting the trace decay rate. We will do better in a revision and would be grateful if the reviewer points us to prior work they are aware of. **Evaluation in small environments.** Unfortunately, many ideas that work well in small environments do not scale to DRL benchmarks. In this work, wefocused on training a single agent across a diverse set of large environments. Self-tuning fits this setup since it allows the agent to adapt differently across environments and time.

R6. Fig 3. As we did for our ATARI experiments, the hyperparameters for IMPALA in Control are the same ones that were used previously in A3C. We use these values for a fair comparison. The hyperparameters that are related to STACX were kept fixed in Atari and Control. It's possible that there are slightly better hyper parameters, but we did not find the algorithm to be too sensitive to them. **Run time.** We would like to refer the reviewer to Table 4 in the supplementary where we discuss this in more detail. In short, the differences in run time are almost negligible since we are not fully utilizing our hardware. Other than that, in many cases the measure of interest is the sample complexity and not the computational complexity, thus, using more compute to gain better performance with under the same sample budget is justified. In other setups, this might not be the case.