We thank all reviewers for their time and comments. Here are some general responses followed by individual ones.

Section A: Related Work. In response to **Reviewer 4**'s interpretation, we'll first contrast our work with Gelada's work and ACE (Imani et al 2019), which will be included in the final version of the paper. Our work relates to Gelada's work by borrowing their covariate shift ($c_{\hat{\gamma}}$). However, how we use $c_{\hat{\gamma}}$ is different. Q-learning is a semi-gradient method and they reweigh the semi-gradient update with $c_{\hat{\gamma}}$ directly. If we would similarly reweigh the policy gradient update in ACE, it would just be an actor-critic analogue of Gelada's Q-learning approach as **Reviewer 1** suggested. However, this reweighed ACE will no longer follow the policy gradient of objective J_μ, yielding instead a "policy semi-gradient". In our work, we define a new objective with $c_{\hat{\gamma}}$ and derive policy gradients for this new objective. The resulting algorithm still belongs to policy gradient methods. However, we then need to deal with $\nabla c_{\hat{\gamma}}$, i.e., compute the policy gradient of a *distribution*. This has not been done in RL and cannot be handled by ACE. ACE is only a special case of our work with $\hat{\gamma} = 0$ where $\nabla c_{\hat{\gamma}}$ disappears. In the on-policy setting, we do not need such gradients due to some algebraic manipulation, which does not work for the off-policy setting. Therefore, ACE uses the sampling distribution d_μ instead of an on-policy distribution to get around this issue. To the best of our knowledge, we are the first to address this issue (computing policy gradients of a distribution) directly with a novel emphatic trace ($F_t^{(2)}$ in our paper). Furthermore, our experiments are much more involved than ACE: Imani et al. evaluated ACE on several handcrafted simple MDPs with linear function approximation, while we scale up both ACE and GeoffPAC to Mujoco with networks.

Section B: TD3. We will include a comparison with TD3 in the next version of the paper as shown by Figure 1. Somewhat surprisingly, TD3 does not work better than DDPG in our setup. We took the TD3 implementation directly from the author's GitHub and report the evaluation performance of the target policy. Using the author's original parameters, in particular an initialization with 10^4 random actions, we reproduced the reported results. However, in our setup all 10^6 samples are drawn from the random sampling policy μ, and in this setting TD3 fails dramatically. This may indicate that TD3 overfits to the common DDPG training setup and emphasizes the difficulty of our experimental setting due to the high degree of off-policy samples.

Section C: Objectives. In contrast to **Reviewer 5**'s interpretation that J_π is similar to J_μ, we will clarify that for off-policy training, the execution of π is imaginary in both objectives. After we run μ till the chain mixes, we continue to run μ, during which time we evaluate $v_\pi(s)$ (using off-policy methods) with states sampled from d_μ. The policy π is therefore never directly executed. Due to function approximation, we cannot maximize $v_\pi(s)$ for all states and have to trade off. J_μ prefers to maximize $v_\pi(s)$ for those states that are often visited by μ, while J_π prefers states that are often visited by π, same as what we prefer in on-policy continuing setting. As state visitation under μ and π can be arbitrarily different, so does J_μ and J_π.

Reviewer 1: **(i)** See Section A. **(ii)** Like Gelada and Bellemare (2019), we use a uniformly random behavior policy to emphasize the importance for correcting the discrepancy between d_π and d_μ. When μ is changed, we may need to change $\hat{\gamma}$ adaptively according to the similarity between π and μ, which we shall investigate in future work.

Reviewer 3: The comparison with TD3 in Section B reveals how much modern OPPG algorithms rely on sufficiently recent on-policy sampling. When the difference between μ and π is large,we would therefore expect GOPPG to improve OPPG algorithms. It would also be possible to include other OPPG improvements into Geoff-PAC, e.g., a V-trace critic or LSTM networks from IMPALA. Additionally, as DDPG often outperforms OffPAC, we would expect a deterministic GeoffPAC to outperform vanilla GeoffPAC as well. We'll connect GOPPG and OPPG more explicitly and investigate GeoffPAC and DDPG with the same architecture and computation resources in the final version of the paper.

Reviewer 4: (Originality) See Section A. **(Computation)** We use a novel emphatic trace ($F_t^{(2)}$ in L194) to estimate $g(s)$ incrementally, which is theoretically supported by Proposition 1. Because we store trajectories in our replay buffer, the sampled data from the buffer can be easily used to compute this trace. We'll clarify this in the final version.

Reviewer 5: (Objectives) See Section C. Furthermore, μ and π are not transient policies before the MDP gets steady. They assign different weights to different states and are never forgotten even after the MDP converges. We will clarify this explicitly in the final version. **(Performance)** The cited paper shows indeed better performance but with a trained expert as the behavior policy. This setup is much easier than sampling from a uniformly random policy μ. As our above comparison with TD3 in Section B demonstrates, off-policy methods are extremely sensitive to the behavior policy, and the two setups are therefore not directly comparable.

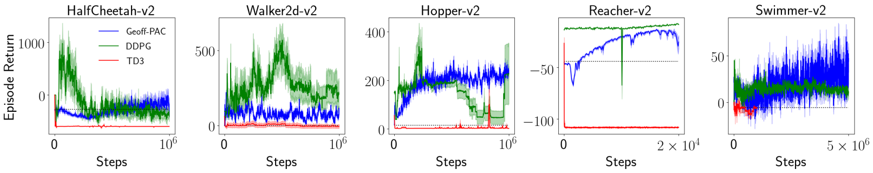

Figure 1: A comparison with TD3. We only run TD3 for 10^6 steps in Swimmer due to time limit. Curves are averaged over 10 random seeds and shadowed regions indicate standard errors. Dashed line is a random policy.