We appreciate the reviewers' thoughtful comments. Due to the space limit, we only address the most salient concerns/questions in the reviews. We will do our best to address the reviewers' other points in the paper's next version.

Global responses to some common concerns among reviewers

Relationship to prior work (R2, R3): While mirror descent has been explored in RL settings before, our paper stands out from all existing work in making three key innovations: (i) using a hybrid representation, a combination of a program selected from a generic programming language and a neural network, for policies; (ii) implementing the projection operator using combinatorial program synthesis; and (iii) a thorough analysis of the consequences of doing gradient descent only in the neural component of hybrid policies, which introduces bias in the gradient operator.

To maintain focus on our main contributions, we used relatively basic versions of the two key components of the IPPG meta-algorithm: mirror descent and program synthesis. However, each of these components can be improved further in future work. In particular, like some of the papers pointed out by R2, one can use extensions of mirror descent such as composite objective mirror descent [Duchi et al., JMLR11]. One can also implement the projection operator using learning-accelerated program synthesis techniques, such as those pointed out by R3. We will include further discussion of these possible directions in the final version.

Incorrect link to code and reproducibility (R1, R3): We most sincerely apologize for the error! We accidentally created two Dropbox folders during submission and uploaded the code to the incorrect folder (due to over-excitement with our paper :-)). We have verified that the code is now available at the included link. We have also provided a Docker container with all dependencies and the TORCS simulator installed, to ease reproducibility.

Responses to other specific questions/concerns

R1 on why projection is easier than direct search in program space: Intuitively, this is because we frame projection as an imitation learning (IL) problem to imitate a neural net, in contrast to the full RL problem of searching over programmatic space. It is widely acknowledged that IL tends to be easier than general RL, which needs to solve planning under possibly long horizon (cf. Deeply AggreVaTeD paper by [Sun et al., ICML18]). This intuition is corroborated by related work on programmatic RL [Verma et al., ICML18], which shows that direct search over programs often fails to meet basic performance objectives (for example having a TORCS car finish a lap in any amount of time).

R1 on our benchmarks and baselines being simple: We respectfully disagree regarding the difficulty level of the TORCS task, our primary benchmark. TORCS has several race tracks, many of which are challenging domains for RL due to the continuous action space and long horizon. TORCS is also among the few standard benchmarks that allow studying generalization properties, i.e., where the training and testing environment can differ. Of the popular policy gradient algorithms, only DDPG has been shown to complete laps in TORCS in the tracks that we considered. Two other algorithms, TRPO and PPO, cannot find policies capable of completing laps in most of these tracks [Cheng et al., ICML19]. In Table 1 in the paper, we show that IPPG learns to drive on some tracks (Ruudskogen and Alpine-2) where even DDPG fails to learn a sensible policy. Thus, we do compare with (and show an improvement over) state-of-the-art deep RL. Finally, we compare IPPG with VIPER and NDPS, the most relevant techniques for programmatic RL.

R3 on assumption for Theorem 4.2: This simplifying assumption is contained to Section 4.2 of the paper, for the purpose of providing a clean finite-sample analysis under vanilla policy gradient. The subsequent Section 4.3 addresses the setting when this assumption does not hold. Our analysis and experiments do not require this assumption in general.

R3 on initial policy: IPPG can be initialized with a neural policy, learned for example via DDPG, and thus can be made to learn "from scratch". This amounts to running the program synthesis projection as a pre-training step (Lines 4-6 in Alg. 1). On most tracks, this gives results comparable to those we have presented. However, note that IPPG with a (very simple) hand-crafted prior can sometimes finish tracks on which DDPG fails. Naturally, this is not possible when we invoke the program synthesis projection during pre-training. Also, using a prior results in "safer" training (Figure 5).

R3 on experimental setup: In principle, IPPG and its theoretical analysis do not depend on the particular policy gradient approach. However, there is a huge empirical difference between different policy gradient algorithms on TORCS. PPO and TRPO actually cannot find policies capable of completing laps in most TORCS tracks we considered (also noted in some previous papers). For this reason, we did not try using them as a component of IPPG in this paper.

R3 on experimental results & reproducibility: We have added a video for the TORCS benchmark, comparing IPPG and NDPS at 4 snapshots during training, to the Dropbox folder for the code. The video shows that IPPG is safer during training, shows better iterative improvements, and recovers more gracefully from crashes than the NDPS agent. The tracks are deterministic, but the initial state (starting position) of the car in each race is chosen randomly from a fixed set. While we did not calculate confidence bounds in the submitted version, a key benefit of hybrid policies (over deep RL) is that they lead to lower-variance learning [Cheng et al., ICML19], and as a result, IPPG is likely to have tighter confidence bounds than DDPG. We will substantiate this point using concrete numbers in the final version of the paper.