Thank you for noting the novelty of LPG-FTW, the fairness of evaluation, and the presentation and strength of results.

R1: Thank you for the very positive comments and detailed suggestions (**requested non-stationary results below**).
3) I.i.d. assumption and non-stationarity: We assume tasks are drawn i.i.d. from a task distribution. In our evaluation, we sample tasks at random, and vary their order across trials (line 249). LPG-FTW supports interleaved MTL, but we evaluate a true lifelong setting, without revisiting tasks. Following your suggestion, we ran LPG-FTW on domains violating this i.i.d. assumption, with gravity increasing linearly in $[0.5, 1.5]g$. LPG-FTW, although not specifically designed to handle non-stationarity, performs equivalently to the i.i.d. setting on Half-Cheetah & Hopper (Fig. 1). We did no parameter tuning due to time; we think Walker2D needs tuning to handle the violated assumption.
3) Storing all Hessians: We don't store $\boldsymbol{H}$'s in practice after updating $\boldsymbol{A}, \boldsymbol{b}$, since LPG-FTW only needs $\boldsymbol{H}$ if revisiting tasks, which we don't do. If tasks are revisited (e.g., interleaved MTL), it would require storing $\boldsymbol{H}$ at a cost of $O(d^2T)$.
8) Poor performance of experience replay (ER): as stated in Sec. 2, ER in PG algorithms (via importance sampling) is unstable. If the policy moves far from earlier tasks, replay stops helping, which is aggravated as training progresses.

R3: Thank you. We address as many comments as space allows, and will update our draft with cites and clarifications.
3&6) Previous work: LPG-FTW improves PG-ELLA in 3 main ways. 1) PG-ELLA trains single-task (STL) policies on each task separately and then finds $\boldsymbol{L}, \boldsymbol{s}^{(t)}$ via dictionary learning on the STL policies (line 60), while LPG-FTW learns directly in the factored space (finding $\boldsymbol{s}^{(t)}$ given the current $\boldsymbol{L}$). 2) PG-ELLA has no explicit initialization, while LPG-FTW uses Algo. 2, ensuring that columns of $\boldsymbol{L}$ are diverse. 3) PG-ELLA assumes STL finds optimal policies, while LPG-FTW adds a linear term to the cost to address non-optimality (line 129). This enables LPG-FTW to work on far more complex tasks than PG-ELLA (Meta-World vs cartpole). We will emphasize these connections in Sec. 4.
3) Chosen baselines: We chose EWC and PG-ELLA as representatives of single- & multi-model classes. Progressive nets scale poorly in lifelong settings. P&C suffers from the same limitations as EWC due to the single-model assumption, and so it wouldn't scale well to the highly diverse tasks we study in Meta-World.
4) Correctness of PG-ELLA evaluation: We consulted with one author of PG-ELLA to validate that our evaluation was correct. In PG-ELLA, the lifelong-learned policies are used only as a warm start for subsequent learning. They first train STL policies, then run dictionary learning on the pre-trained policies to find an $\boldsymbol{L}$ matrix, and finally use the $\boldsymbol{L}\boldsymbol{s}^{(t)}$ policies to start a second STL process—for evaluation. Their main result (Fig. 2 in PG-ELLA) shows that initializing STL from a lifelong-learned policy accelerates training. In contrast, our evaluation is entirely lifelong learning: the agents are evaluated as they train on each task sequentially. There, PG-ELLA *does not* leverage information across tasks (Fig. 1 and 3-top in our paper). Once all tasks are trained via STL, PG-ELLA runs dictionary learning and we evaluate the policy in bars 3 and 4 of Fig. 2 and 3-bottom. Bar 4 corresponds to the starting point of the evaluation in PG-ELLA.
5) Clarity of notation: LPG-FTW keeps <u>one</u> dictionary shared across tasks: the multi-task cost in Eq. 2 is optimized incrementally with auxiliary matrices $\boldsymbol{A}, \boldsymbol{b}$. $\boldsymbol{L}_t$ denotes the $\boldsymbol{L}$ learned up to task t. We only keep one $\boldsymbol{A}$ matrix for all tasks ($\boldsymbol{A}_t$ is an auxiliary variable). $\hat{t}$ indexes over previous tasks, and t is the current task. $\boldsymbol{\alpha}$ is the policy obtained for a task after its training process (using the $\boldsymbol{L}$ *up to the previous task*). We will emphasize these clarifications in the text.
8) Additional experience initializing L?: There is none: initialization replaces the standard training for tasks 1–k.
8) Why is STL=PG-ELLA in 5.1 but not 5.2?: Figs. 1 and 3-top show the training process, for which PG-ELLA uses STL. Figs. 2 and 3-bottom show subsequent steps (bars 3 and 4) where PG-ELLA combines information from all tasks.
8) Comparable capacity?: All methods used policies of the same size, but overall capacity naturally varied across methods. We followed your suggestion of increasing the capacity of EWC to match that of LPG-FTW in Meta-World, and found that results didn't change significantly (Fig. 2).
7) Reproducibility: The clarifications above, along with the full code we provided, should make our results reproducible.
8) How does training progress?: Number of iterations is summarized Appendix B. Each task is seen by the agent only once, and there is no possibility of going back for further experience at any point during the process.
8) EWC performance drop after training all tasks: EWC suffers from catastrophic forgetting, unlike our approach.

R4: Thank you for the very positive comments. For connections to PG-ELLA, see **R3**.
8.1) Connections between charts: 'Tune' matches the end of the curves, considering only per-task training but not how that affects previous tasks. 'Update' and 'Final' come later: 'Final' assesses performance after all tasks are trained.
8.2) Does STL performance imply tasks are similar?: STL *ignores* all other tasks when training on one task, so task similarity does not play a role. STL performs well because STL performance was used in the design of the benchmarks.

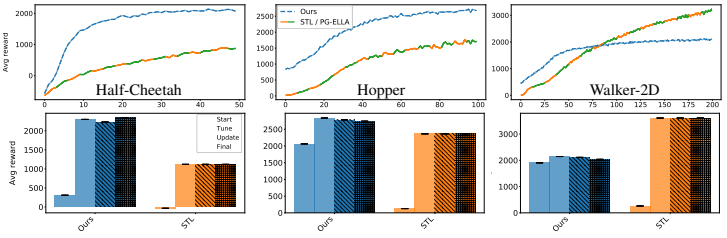

Fig. 1: Non-stationary results requested by **R1**. STL performance (for reference) measured on original i.i.d. task distribution.

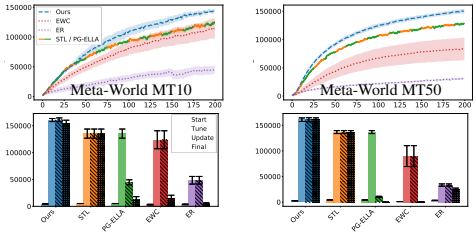

Fig. 2: Higher EWC capacity, requested by **R3**.