To Reviewer #2: Thanks for recognising the novelty, clarity and promising results of our work.

Reproducibility: Please note that our supplemental material contains both the **complete architecture** of the agent and **detailed pseudo-code** for the algorithm. We'll add a reference to the appendix in the main text of the paper.

Catch related: We'll provide more details on the learning to bootstrap results. Note that the agent only sees a limited window of 3 time-steps, and hence cannot see all the way to the end of the episode to provide the full return. For your specific questions, 1) the 3-step look-ahead baseline is the theoretical optimal performance any agent can achieve without bootstrapping. If an agent performs better than this score, it shows that the algorithm has learned how to bootstrap. We'll provide detailed calculation of the theoretical optimal performance in the appendix; 2) We'll add an RL baseline for the Catch environment.

Robustness of learning a target vs. learning a loss: For any given η the expected target $\mathbb{E}(g_\eta(\tau))$ is a well-defined function, even if the target jumps around "haphazardly" from state-to-state or step-to-step. Optimising a squared loss by SGD simply adjusts the value function $v_\theta(S)$ in a way that minimises MSE to the expected target. In contrast a learned loss could yield arbitrary dynamics (for example, moving *away* from a target) which might then diverge easily. We have observed this in our experiments. E.g., Figure 3b shows that learning a loss was more fragile and the agent failed to achieve good performance on large-scale experiments.

To Reviewer #3: We appreciate your valuable comments! We will add a baseline for [1].

To Reviewer #4: Thanks for noting that our paper tackles a very valuable problem and that the idea is interesting & novel.

Trajectory encoder as additional input to value and policy:

The "5-state random walk" was designed to demonstrate FRODO can learn to handle non-stationarity. You are correct that a memory-based solution could in principle also deal with non-stationarity; however this is typically difficult when the time-scales are long (e.g. there are hundreds of steps between switches in the random walk example). Following your suggestion, we ran TD(λ) using an LSTM to encode the history. Results are shown in the Figure 1 of the rebuttal, and the error metric is mean over 10 random seeds. FRODO performs better than this new baseline, suggesting it deals more effectively with non-stationarity.

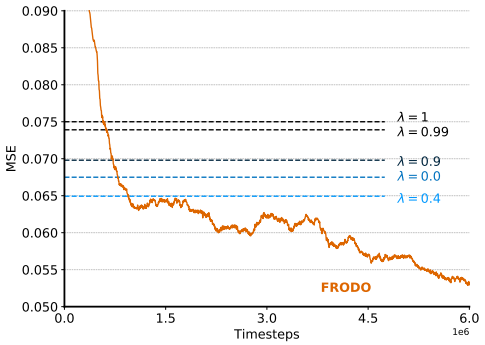

Figure 1: Comparison of FRODO with trajectory encoded value prediction in "5-state random walk" environment (suggested by Reviewer #4).

"Catch" was used to show that FRODO can learn to bootstrap when given only a window of 3 time-steps into the future. Adding a trajectory history (e.g. LSTM) as input to the value function does not help with the issue of bootstrapping in Catch, and will result in the same theoretical upper bound as shown in the paper (a low score around 0.2).

Comparisons to baselines: Our goal is an algorithm that learns its own objective solely from experience of interacting with its environment. The comparison to IMPALA is to show that the proposed FRODO algorithm can learn its own objective to outperform a strong actor-critic baseline on challenging domains. Note that FRODO algorithm uses exactly the same outer loss as the IMPALA baseline. FRODO builds upon the IMPALA baseline and furthermore learns to outperform it eventually. Interestingly from our analysis, FRODO learns a very different objective from IMPALA. We'd like to point out that outperforming an well-established RL algorithm by learning discovering RL objective online is non-trivial and to our knowledge this work is the first one to achieve such strong performance.

As for stronger baselines in Atari such as NGU, Agent57, and MuZero, the main goal in the paper is not to achieve state-of-the-art performance in specific domain, but to introduce a general algorithm which learns its own objective online. Using IMPALA as outer loss is clean and simple to show the effectiveness of our proposed algorithm. As we've mentioned in the paper, any RL objective can be used as outer loss in the proposed FRODO algorithm. People can easily extend and apply our FRODO algorithms with more complicated RL algorithms like NGU and Agent57.

Implementation and code: Please note that we provided **full pseudocode** in Python format in the appendix.

Trajectory length in τ: Trajectory τ is a truncated-length trajectory within a single episode. It is with length 10 in the "5-state random walk" experiment.