A. Hyperparameters

Table 1. PPG-Specific Hyperparameters

N_π	32
E_π	1
E_V	1
E_{aux}	6
β_{clone}	1
# MINIBATCHES PER AUX EPOCH PER N_π	16

Table 2. Other Hyperparameters

γ	.999
λ	.95
# TIMESTEPS PER ROLLOUT	256
# MINIBATCHES PER EPOCH	8
ENTROPY BONUS COEFFICIENT (β_S)	.01
PPO CLIP RANGE (ϵ)	.2
REWARD NORMALIZATION?	YES
LEARNING RATE	5×10^{-4}
# WORKERS	4
# ENVIRONMENTS PER WORKER	64
TOTAL TIMESTEPS	100M
LSTM?	NO
FRAME STACK?	NO

We used the Adam optimizer (Kingma & Ba, 2014) in all experiments. We used a combination of NVIDIA V100 and NVIDIA P100 GPUs.

We normalized rewards so that time-discounted returns had approximately unit variance. Specifically, we tracked an approximation of the standard deviation of time-discounted returns and, at each time step, divided rewards by this standard deviation. See code at https://github.com/openai/phasic-policy-gradient.

B. Shared vs Separate Networks

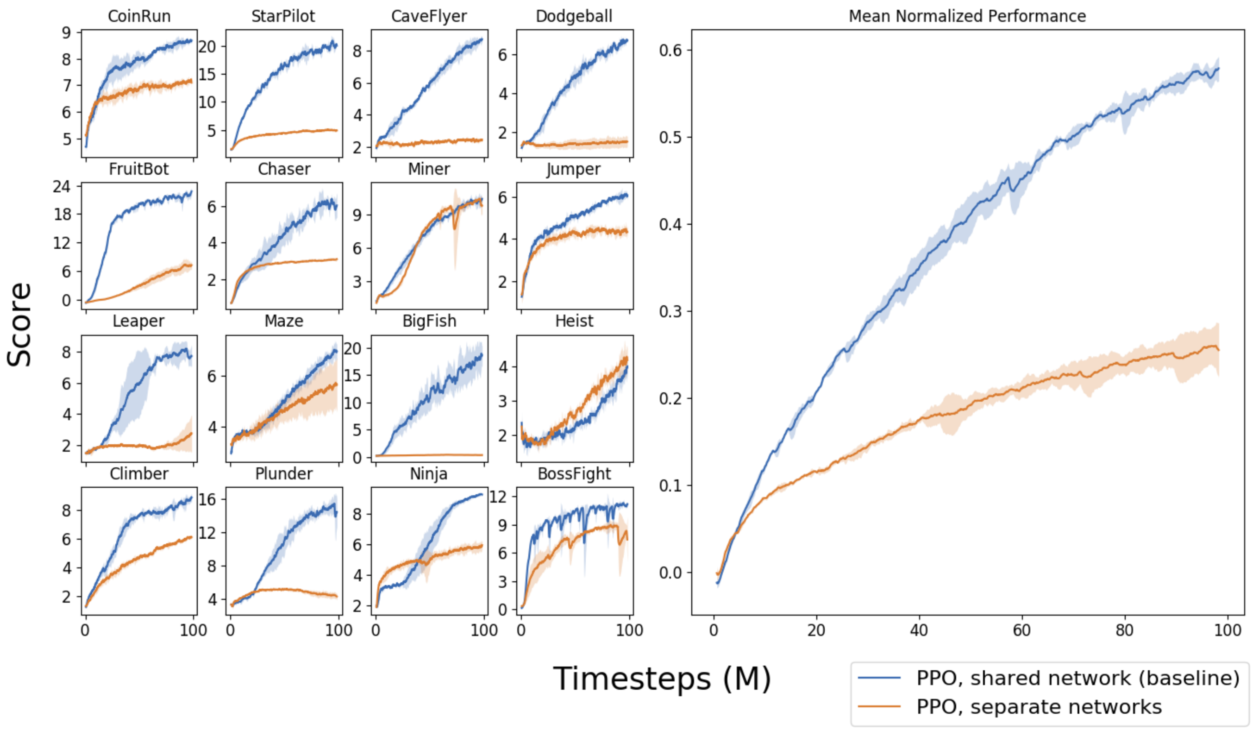

Figure 8. A comparison between two implementations of PPO on Procgen Benchmark. The baseline shares features between the policy and value networks, while the ablation trains separate policy and value networks.

C. Auxiliary Phase Value Function Training

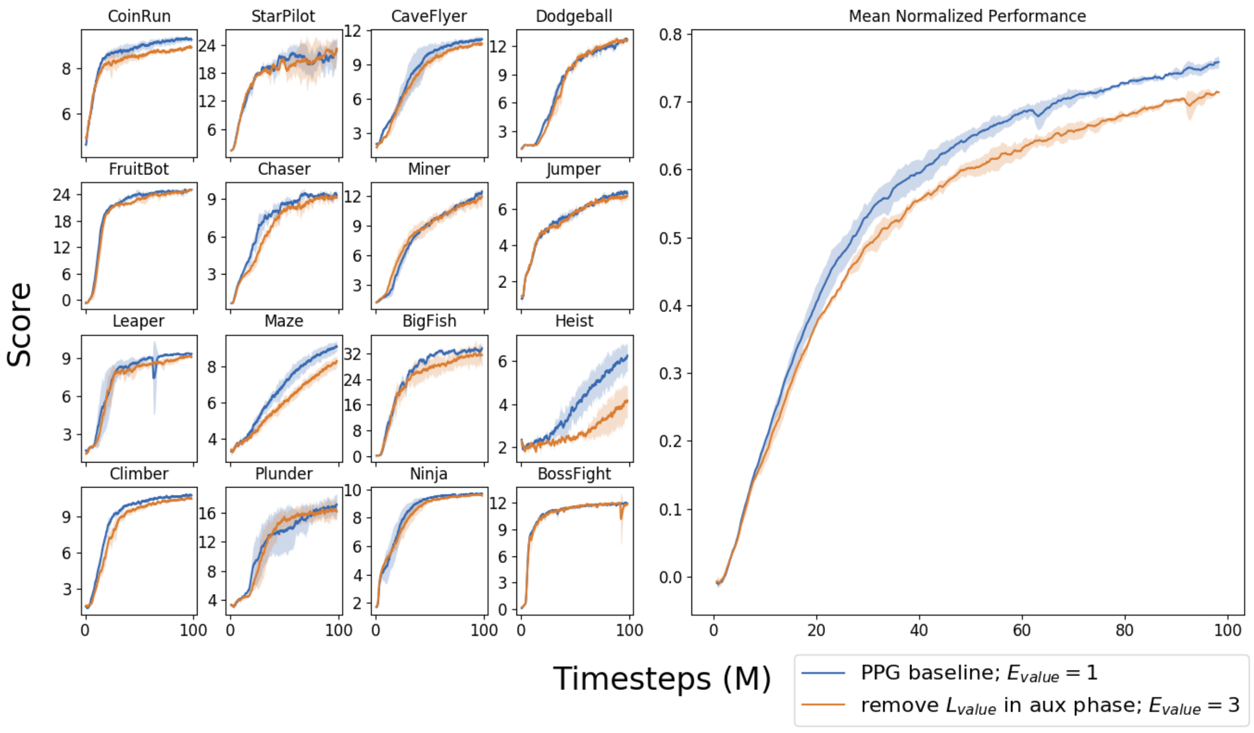

Figure 9. The performance of a variant of PPG which skips the optimization of L^{value} during the auxiliary phase, in favor of additional optimization of L^{value} during the policy phase.

We now discuss the relative importance of optimizing L^{value} and L^{joint} during the auxiliary phase. From Appendix B, we know that L^{joint} is crucial; without some optimization of this objective, there is no mechanism to share features between the value function and the policy. Although it is convenient to optimize L^{value} during the auxiliary phase as well, it is not strictly necessary. It is also viable to perform extra value function optimization during the policy phase (by increasing E_V), while removing the optimization of L^{value} from the auxiliary phase. A comparison between this variant and the PPG baseline are shown in Figure 9. Although the PPG baseline has a slight advantage, we can see that the choice to optimize L^{value} during the auxiliary phase is not an essential element of PPG.

D. PPO Sample Reuse

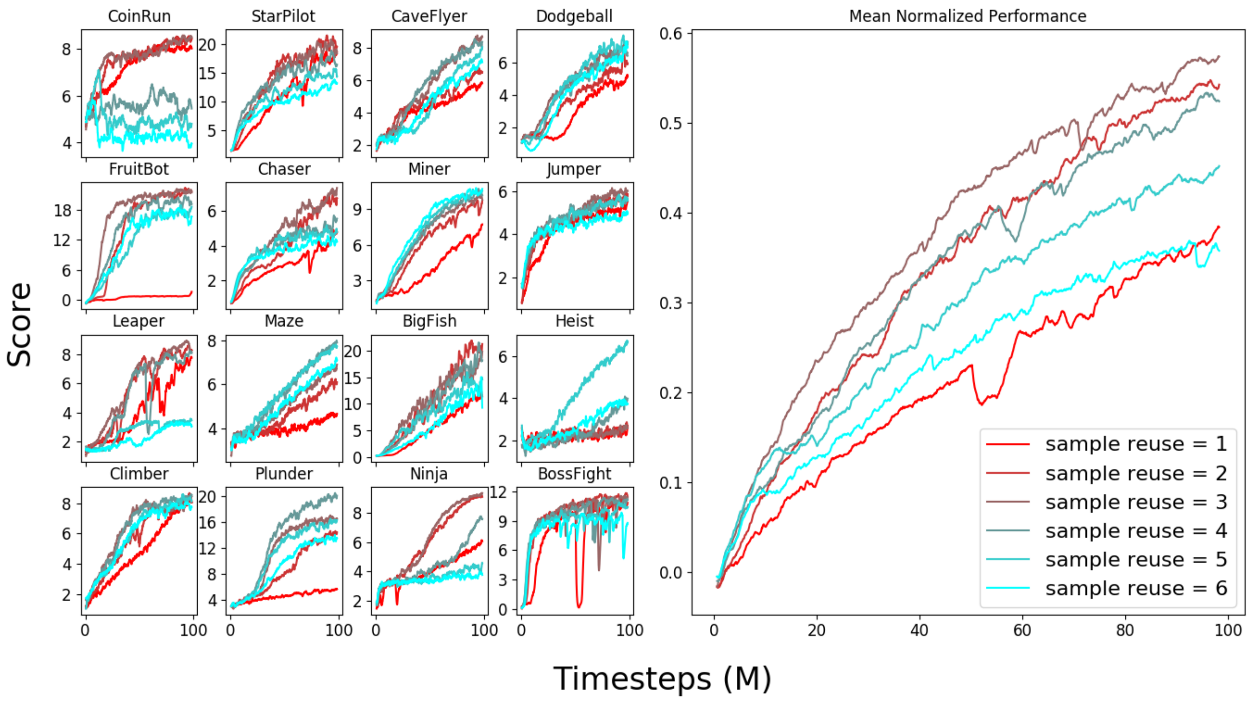

Figure 10. A comparison between different levels of sample reuse in PPO.

We sweep over the different values for sample reuse in PPO, from 1 to 6. Empirically, we find that a sample reuse of 3 is optimal, given our other hyperparameter settings. As discussed in Section 3.2, the results with PPG suggest that the poor performance of PPO with low sample reuse is due to the fact that the value function, not the policy, is being under-trained.

E. Additional Breakdowns by Environment

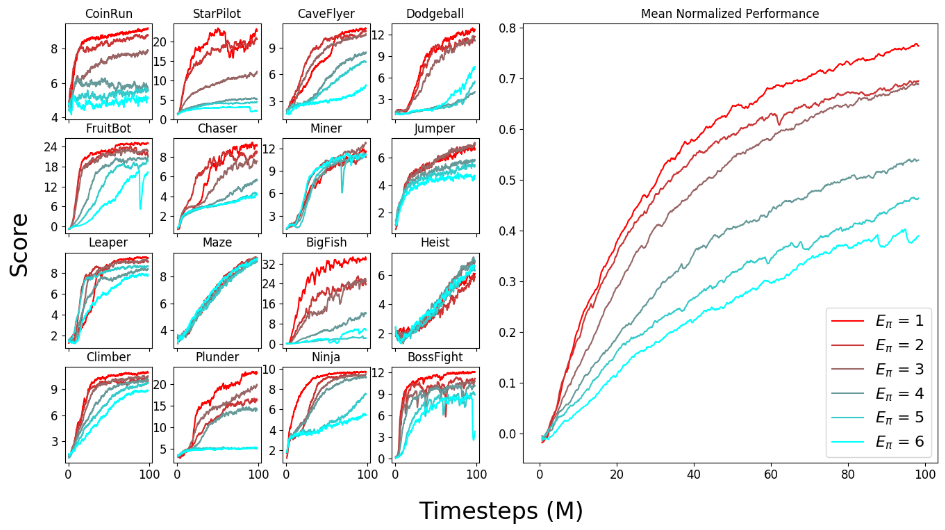

Figure 11. An environment specific breakdown of Figure 3. Performance with varying levels of policy sample reuse.

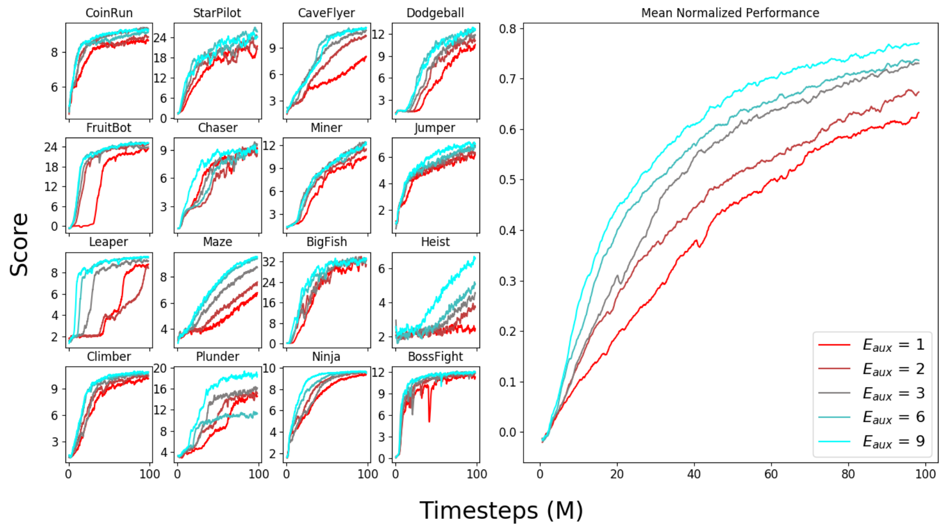

Figure 12. An environment specific breakdown of Figure 4. Performance with varying levels of value function sample reuse.

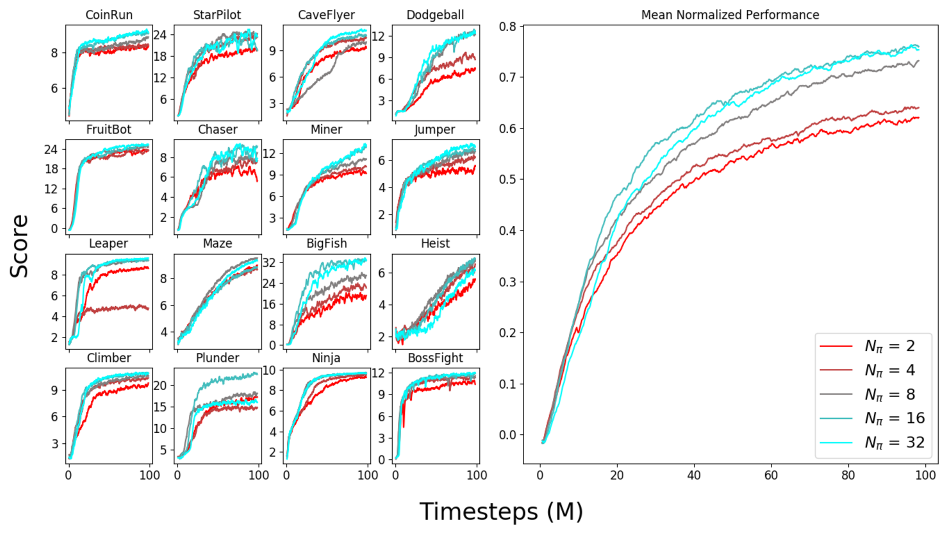

Figure 13. An environment specific breakdown of Figure 5. Performance with varying auxiliary phase frequency

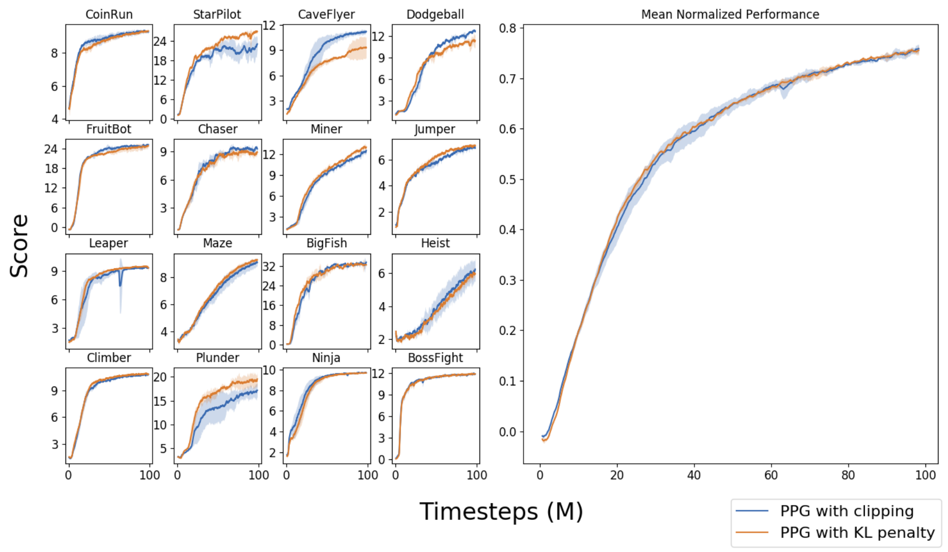

Figure 14. An environment specific breakdown of Figure 6. The impact of replacing the clipping objective (L^{clip}) with a fixed KL penalty objective (L^{KL})

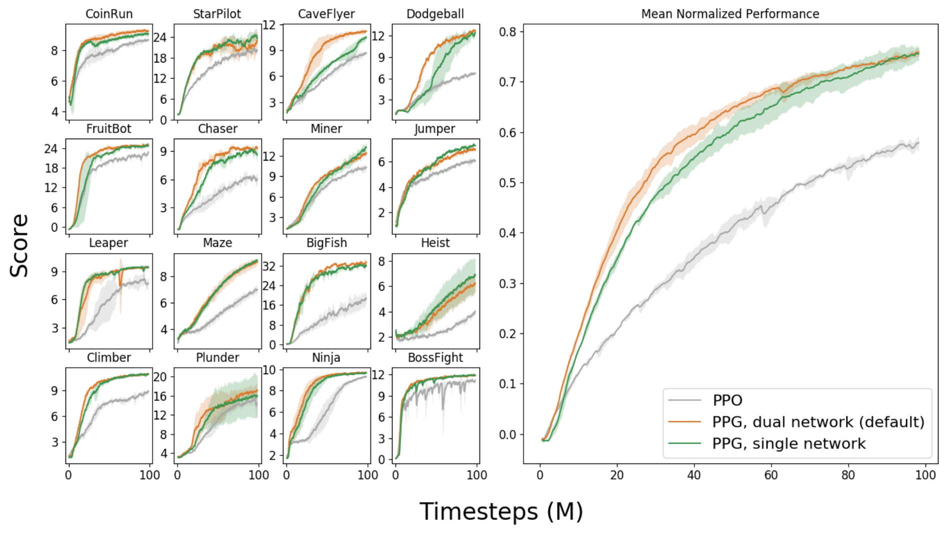

Figure 15. An environment specific breakdown of Figure 7. A comparison between the default implementation of PPG which trains two separate networks, and a single-network variant that mimics the same training dynamics by detaching the gradient when necessary. PPO shown for reference.