

Is Your Auxiliary Loss Still On? Measuring Gradient Liveness in PPO

Abhisek Keshari
Independent researcher

abhisek.keshari12@gmail.com

Abstract

Auxiliary self-supervised losses in reinforcement learning are judged by the final return of the agent that carries them, which cannot tell an objective that is still supplying gradient from one whose gradient has faded. We log the auxiliary loss’s gradient at the shared encoder over training, at the cost of two extra backward passes, and find that one of three objectives supplies a gradient that declines through the whole run. With PPO on three ProcGen games, SPR’s moving-average target saturates within the first 6.3% of training, and its auxiliary gradient then declines for the rest of the run, ending at a median 0.039 of its early value against 0.60 for CURL and 0.79 for reconstruction; the groups overlap at one run on this primary statistic. The decline is in SPR’s own gradient, not in a growing PPO gradient. Across a sweep of the target’s momentum, saturation comes later at higher momentum but sub-linearly in the target’s time constant, against our pre-set prediction. A retrospective 500K-step check on SPR’s own loss flagged 8 of SPR’s 9 runs. Re-scored under the companion protocol with a measured random floor, no objective’s effect on return survives correction.

1 Introduction

This paper began as a different study. We set out to test whether an auxiliary self-supervised loss helps a PPO agent that appears to have stopped improving on ProcGen, and to see what each loss was doing we logged its gradient at the shared encoder.¹ One objective’s auxiliary gradient declined steadily over the whole run, in every run, and ended far below where it started. This paper characterises that decline: which objective it affects, where in the gradient it sits, and what it tracks.

Auxiliary losses in deep reinforcement learning come in three families. A contrastive objective, here CURL (Laskin et al., 2020), pulls two views of the same observation together. A predictive objective, here SPR (Schwarzer et al., 2021), predicts the encoder’s own future latent state against a slowly moving target. A reconstruction objective decodes the observation from the encoder’s features under a pixel loss. All three are attached to a shared encoder, trained alongside the policy, and judged by the return of the resulting agent. But an auxiliary loss acts on the encoder only through its gradient. If that gradient shrinks to almost nothing over the run, the encoder spends most of training shaped by the reinforcement-learning objective alone, and the final return describes an agent the auxiliary loss barely touched. The usual evaluation cannot tell.

In companion work we showed that ProcGen evaluation numbers need a measured random floor and an explicit action rule before they can be read (Keshari, 2026). Here we ask the analogous question of the training signal: is an auxiliary objective still contributing gradient when we measure its effect?

With PPO on three ProcGen games and three objectives, we find:

1. **SPR’s auxiliary gradient declines steadily; CURL’s and reconstruction’s persist.** At the end of training SPR’s auxiliary gradient is a median 0.039 of its early value, about 16 times below CURL’s ratio, and reconstruction’s is higher still (Section 3).

¹The author used AI assistants (Claude) for code implementation, analysis scripting and editing of the manuscript. The research questions, experimental design, analysis plan, interpretation and all claims are the author’s; every reported number was produced by, and checked against, the analysis code.

2. **The collapse is in the auxiliary gradient itself**, not in a growing PPO gradient. Splitting the auxiliary share of the encoder gradient into its two parts attributes a median 0.75 of SPR’s decline to its own gradient (Section 4).
3. **It tracks saturation of SPR’s moving-average target**. The target saturates within the first 6.3% of training in every run, and across a sweep of the target’s momentum, saturation comes later at higher momentum but sub-linearly in the target’s time constant, so our pre-set prediction failed (Section 5).

Section 6 asks whether a live gradient changes the policy, scored with the companion paper’s protocol; at these seed counts no objective’s effect survives correction. Section 7 gives a short check for the target’s saturation, and the study’s limits.

Related work. Auxiliary losses entered deep RL with UNREAL (Jaderberg et al., 2017); CURL (Laskin et al., 2020) adds an InfoNCE loss over augmented crops and SPR (Schwarzer et al., 2021) predicts future latents through a momentum target. They are evaluated by return, as is the closest negative result, that joint self-supervised training does not beat augmentation alone at matched data (Li et al., 2022). Theory shows that EMA or stop-gradient targets avoid trivial collapse under idealised assumptions (Tang et al., 2023; Ni et al., 2024), and analyses when latent self-prediction helps relative to reconstruction (Voelcker et al., 2024); an objective can avoid collapse and still stop supplying gradient. PPO agents lose plasticity and representation quality over training (Lyle et al., 2022; Sokar et al., 2023; Dohare et al., 2024; Juliani & Ash, 2024; Moalla et al., 2024), and PPO plateaus can come from degraded gradient estimates at low parallelism (Beukman et al., 2026). Those accounts concern a network that no longer responds, or an estimate that has degraded; gradient liveness concerns a signal that no longer arrives. An absent gradient rules out the other accounts for that objective, but a present one does not rule them in. Multi-task learning reads interference and balance off per-task gradients (Yu et al., 2020; Chen et al., 2018); our quantity is one such magnitude, read at intervals and never acted on.

2 Setup and measurement

Agent and games. PPO with the IMPALA convolutional encoder (Espeholt et al., 2018), shared by a linear policy head and a linear value head, on coinrun, heist and miner (Cobbe et al., 2020) in easy mode, for 8M environment steps per run and 3 seeds per condition. Every other setting is identical across conditions and listed in Appendix A; there is no reward normalization or clipping.

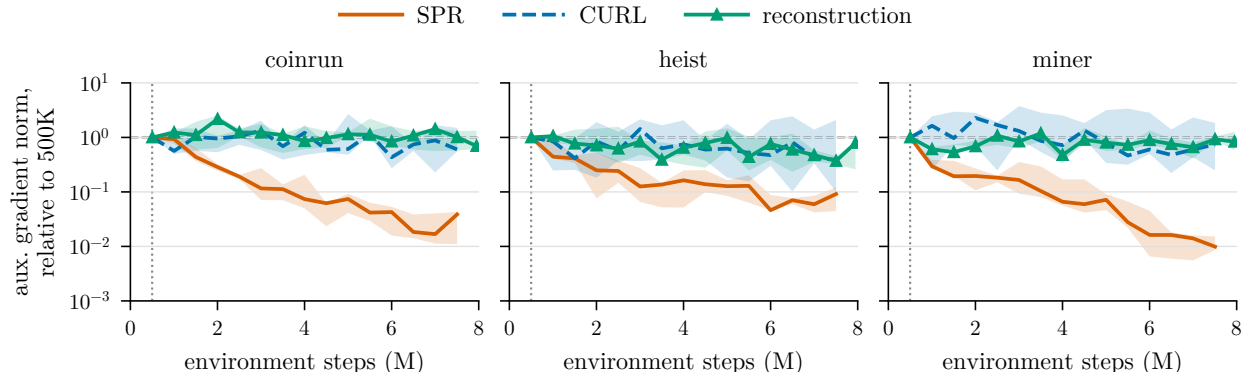


Figure 1: SPR’s auxiliary gradient declines steadily through training in every game, while CURL’s and reconstruction’s hold. Auxiliary gradient norm at the encoder relative to its value at 500K steps (dotted line), median over 3 seeds with the min–max band, log scale, one panel per game; the dashed line marks no change. Reconstruction, logged more often, is shown on the other objectives’ 500K grid with its terminal point kept; every run at its native cadence is in Figure 4.

```

enc = [p for p in agent.encoder.parameters() if p.requires_grad]
g_ppo = grad(ppo_loss, enc, retain_graph=True, allow_unused=True)
g_aux = grad(aux_loss, enc, retain_graph=True, allow_unused=True)
A, P = norm(g_aux), norm(g_ppo)
share = A / (A + P)

```

Listing 1: The hook. Two extra backward passes over the encoder per logged minibatch; the update is unchanged.

Objectives. Three auxiliary losses share the encoder with the policy, each with its own coefficient (Appendix A); the baseline has none.

- **CURL** (contrastive) (Laskin et al., 2020): an InfoNCE loss over two random crops of the same observation.
- **SPR** (predictive) (Schwarzer et al., 2021): predicts the encoder’s latent state 3 steps ahead, scored by cosine similarity against a target encoder that is an exponential moving average of the online encoder with rate $\tau = 0.99$ per PPO update (Section 5).
- **Reconstruction**: decodes the observation from the encoder features under a pixel loss.

The hook. Listing 1 is the whole measurement: five lines in the training loop, run on one minibatch per logging interval, before gradient clipping. `torch.autograd.grad` returns the gradients as values without writing them into any parameter’s `.grad`, so the optimizer step that follows sees exactly what it would have seen without the hook; `retain_graph` keeps the graph alive for the second call and for the real backward pass, and `allow_unused` returns `None` for an encoder parameter a loss does not reach. The hook creates no tensor the update uses and consumes no random numbers; two seed-matched runs with and without it produced byte-identical weights at every checkpoint (Appendix H). Clipping multiplies every gradient by one common factor, which cancels in the share. Per-task gradient norms at shared parameters are a standard quantity in multitask learning (Chen et al., 2018); we read one over training and act on nothing.

The quantities. On one minibatch, A is the norm of the auxiliary loss’s gradient with respect to the encoder parameters, after its coefficient is applied, and P is the same for the PPO loss. The auxiliary share is $s = A/(A + P)$, the fraction of the encoder gradient that the auxiliary loss supplies.

Ratios. We summarise a run by the terminal-over-early ratio of A , $R_{\text{aux}} = A_{\text{term}}/A_{\text{early}}$, where early is the logged step nearest 500K (503,808 in every run) and terminal is the last logged step. The ratio normalises each run by itself, so the loss coefficient cancels from it exactly, and the 500K anchor sits after the initialisation transient. The same ratio of the share, R_{share} , asks whether the auxiliary gradient still matters relative to PPO’s own. R_{aux} is the primary statistic and R_{share} the secondary.

What came first. The SPR and CURL runs were trained first, the decline in Figure 1 was seen in them, and the reconstruction runs were added afterwards. Two tests were then fixed before their data existed: the momentum sweep’s prediction and acceptance window (Section 5), and the decision rules under which Section 6 re-scores every checkpoint. The wording rule for Section 3’s comparison (medians, ranges and the count of overlapping runs) was fixed before those numbers were computed. Those are the confirmatory parts; everything else, including the probe of Section 7, is descriptive. Documents and hashes are in the supplementary files.

3 SPR’s gradient falls; CURL’s and reconstruction’s hold

SPR’s auxiliary gradient ends at a median 0.039 of its early value (Table 1). CURL’s and reconstruction’s end at medians 16 and 21 times higher, near their early values. Figure 1 shows the shape behind the ratios:

Table 1: Terminal-over-early ratio of the auxiliary gradient norm (R_{aux} , primary) and of the share (R_{share} , secondary): median (range) over each objective’s runs, and the number of runs at or below SPR’s largest R_{aux} . The two statistics disagree at the edge, one CURL run lying at or below SPR’s largest R_{aux} while on R_{share} no run of another objective does, and both are reported. Shuffled CURL, a control, is in Appendix E.

Objective	R_{aux} median (range)	R_{share} median (range)	n	At or below SPR’s largest R_{aux}
SPR	0.039 (0.009–0.121)	0.029 (0.003–0.097)	9	–
CURL	0.60 (0.10–2.10)	0.454 (0.149–0.903)	9	1
reconstruction	0.79 (0.26–1.33)	0.601 (0.143–0.910)	9	0

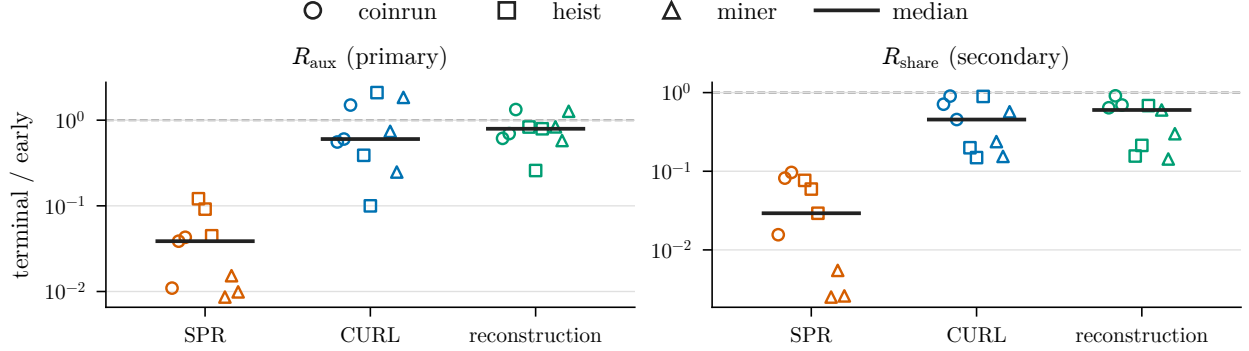


Figure 2: Terminal-over-early ratio of the auxiliary gradient norm (R_{aux} , left) and of the share (R_{share} , right), one point per run, marker by game, median as a bar, dashed line at no change. The two statistics disagree at one run: on R_{aux} one CURL run lies at or below SPR’s largest, on R_{share} none does. Shuffled CURL, a control, is in Appendix E.

SPR’s trajectories decline steadily from the anchor to the end of training, while CURL’s and reconstruction’s wander within a factor of a few of where they started.

The ranges touch at one run. One CURL run ends at 0.100, at or below SPR’s highest ratio (Table 1); every other CURL and reconstruction run ends above every SPR run (Figure 2). On the share ratio the groups do not touch at all. The share is bounded by one, and that CURL run’s early share was 0.81. Its A/P fell to 0.032 of its early value, so a large fall in A/P became a smaller fall in the share (Appendix B). On the ratio of A/P itself, 2 runs sit at or below SPR’s largest.

Shape, not size. Persistence describes the shape of a trajectory, not its size. Reconstruction’s auxiliary gradient holds while being 14–25 times smaller than CURL’s. CURL’s never fell below about 0.19 at any logged step of any run. Whether an objective keeps contributing and how much it contributes are different questions.

4 The decline is in the auxiliary gradient

A share can fall because the auxiliary gradient A falls or because PPO’s gradient P rises, and the two are separable exactly. Since $s/(1-s) = A/P$, the change in the share’s log-odds between the early and terminal measurements splits as

$$\Delta \log \frac{s}{1-s} = \Delta \log A - \Delta \log P,$$

where Δ is terminal minus early, and the fraction of the decline carried by the auxiliary gradient is $f_{\text{num}} = \Delta \log A / \Delta \log \frac{s}{1-s}$. Across SPR’s runs f_{num} has a median of 0.75 and is never below 0.67 (Appendix D): the auxiliary gradient itself carries most of the decline. A value above one means P fell as well, so the share understates the fall in A .

PPO’s own encoder gradient follows the game, not the objective, in its trend. In every arm its terminal-over-early ratio has the direction and order of magnitude of the other arms in that game, with per-game medians over arms from 0.93 on coinrun to 4.82 on miner (Figure 5). Its level is another matter: CURL’s P runs higher than the other arms’ on heist and miner, a difference this study does not test and the decomposition does not use, since it compares two measurements within one run. The split uses the same two measurements as R_{aux} , so it describes those two points, not the whole trajectory.

5 Mechanism: EMA target saturation

SPR scores its prediction by cosine similarity against a target encoder that is an exponential moving average (EMA) of the online encoder. After every PPO update,

$$\theta_{\text{target}} \leftarrow \tau \theta_{\text{target}} + (1 - \tau) \theta_{\text{online}},$$

with the EMA rate τ close to one. This is the “momentum encoder” of MoCo (He et al., 2020), which BYOL (Grill et al., 2020) uses as its target network; the term refers to τ , and has nothing to do with the optimizer’s momentum. When the prediction matches that slowly moving copy almost exactly, the loss sits near its ceiling and supplies little gradient. We call the target *saturated* at the first update after which the cosine stays at or above 0.99 to the end of the run, and a condition’s *median saturation* is the median of that update over its seeds.

At $\tau = 0.99$ the target saturates within the first 6.3% of training in every core run (Table 2). Shorter re-runs on miner, which reproduce the core runs exactly and log every update, put it at updates 66–92. Changing the prediction horizon does not delay it: in a separate shorter study, every horizon saturated inside the same early window (Table 2).

Table 2: When SPR’s target saturates: the first update after which its cosine stays at or above 0.99 to the end of the run.

Runs	n	Saturation
Core runs, $\tau = 0.99$, cosine logged every 50 updates	9	by step 503,808 (6.3% of training)
Re-runs on miner, 1M steps, cosine logged every update	3	updates 66–92 (270K–377K steps, 3.4–4.7%)
Horizon study, prediction horizons of 3, 5 and 10 steps	3	327,680–368,640 steps

Momentum sweep. A moving average with rate τ forgets its past over roughly $1/(1 - \tau)$ updates, so we predicted saturation after about that many updates, and fixed in advance the window of log–log slopes that would count as agreement. We ran SPR on miner at five values of τ (Table 3). Saturation came later at higher target momentum, rising at every step of τ from a median of 20 to 139 updates with no overlap between neighbouring per-seed ranges, but it rose sub-linearly in the time constant. The log–log slope is 0.63, below the pre-specified window, so the prediction failed (Figure 3; the rule is in Appendix G). Low momentum saturated later than predicted and high momentum earlier, consistent with, though not a test of, a minimum time the online encoder needs to settle.

Table 3: The momentum sweep on miner, 3 seeds per value of τ : predicted saturation $1/(1 - \tau)$, the per-seed range and the median, in updates.

τ	Predicted (updates)	Saturation, per seed	Median
0.90	10	18–22	20
0.95	20	23–35	24
0.98	50	38–51	47
0.99	100	66–92	89
0.995	200	102–140	139

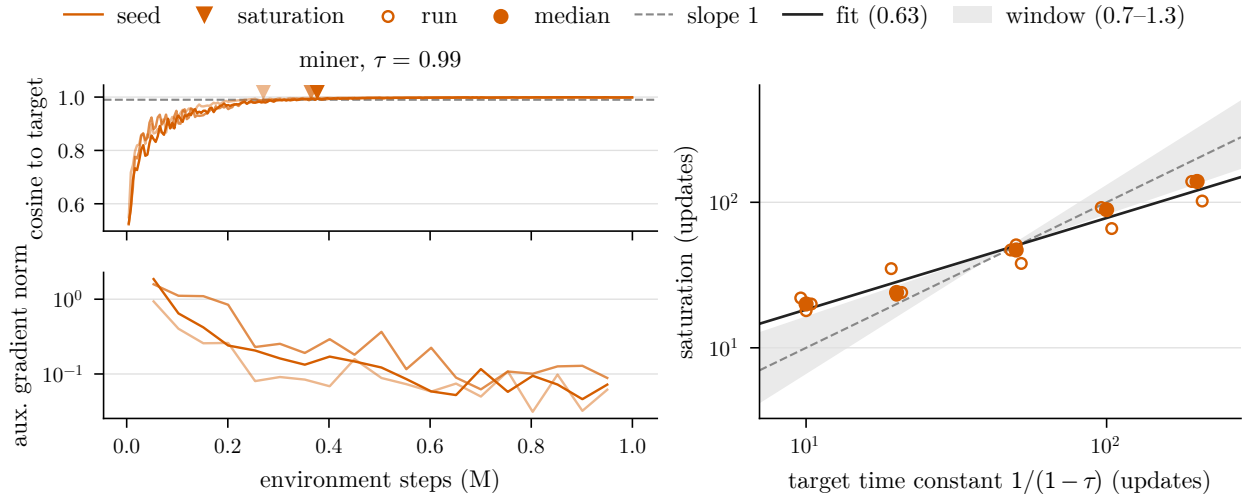


Figure 3: Saturation comes later at higher momentum, but sub-linearly in the time constant: the fit falls below the pre-specified window. Left: the three 1M-step re-runs at $\tau = 0.99$ on miner, one line per seed, logged every update: the cosine to the target with the saturation threshold and each seed’s saturation update marked (top), and the auxiliary gradient norm (bottom). Right: saturation update against $1/(1-\tau)$, log-log, hollow markers per run and filled medians per τ , with the slope-one line, the fitted line and the pre-specified slope window shaded.

What this does not establish. It does not show that saturation causes the collapse: the plan’s secondary check, reported in Appendix G, starts at saturation and so cannot show order. It does not say whether the encoder solved the prediction task or collapsed onto an easy one. And SPR is the only objective here with a bounded, saturating loss, so a bounded loss and a self-generated target remain equally good explanations (Section 7).

6 Does it change the policy?

We scored every final checkpoint with the companion paper’s protocol (Keshari, 2026): sampled actions as the primary rule, three seeded level draws on training and held-out levels, the uniform-random floor on the same draws, and Welch tests with Holm correction across the 9 objective-versus-baseline contrasts (Table 7). No objective’s effect on sampled held-out return, sampled training return or merged-action entropy survives correction; the largest difference, CURL on miner, is +1.93 return units (Holm-adjusted $p = 0.156$), and the three seeds added under a fixed stopping rule show the same sign on their own. On heist the baseline, SPR and reconstruction score below a uniform-random policy on held-out levels and CURL cannot be told apart from it, so heist’s contrasts compare policies at or below chance and say little about transfer. We make no claim that any objective has no effect, and none that a live gradient improves the policy.

7 Practical use and limitations

Practical use. A short run can tell whether SPR’s target has saturated before a long one is paid for. A probe of 500K steps costs 9.4 minutes on the GPU this study used, and asks only whether SPR’s own cosine has reached its ceiling. It flags 8 of SPR’s 9 runs (Appendix F). It is a check, not a result, and it has three limits. It keys on the objective’s own loss, which works for SPR because a cosine has a ceiling; CURL and reconstruction have none, and no probe is stated for them. Its false-positive rate is unmeasured, because the check cannot be evaluated on an objective without a ceiling. And its hit rate is measured on the runs it was designed from. A flagged run means the experiment stopped testing the objective, not that the idea behind it failed; the response is to fix the target or its schedule and measure again.

Limitations. One algorithm, one encoder, three games in easy mode, 3 seeds per condition, one implementation of SPR. The finding is post hoc in origin: the decline was seen in the first runs trained, and only the momentum sweep’s prediction, the decision rules of Section 6 and the wording rule for Section 3’s comparison were fixed before their data existed. The plateau premise behind the original question was tested with one 25M-step baseline run per game, was unresolved, and nothing here tests a plateaued agent.

SPR is the only objective here whose gradient falls this far in every run, but the comparison is not clean at its edge: one CURL run ends at 0.100, below SPR’s largest ratio of 0.121. SPR is also the only objective whose target is produced by the network being trained and the only one whose loss has a ceiling, so CURL and reconstruction have no saturation analogue and the study cannot separate a bounded loss from a self-generated target; a momentum target scored by an unbounded regression loss, or a second self-predictive objective such as MLR (Yu et al., 2022) or BYOL-Explore (Guo et al., 2022), would. The early anchor is fixed at 500K steps, and no sensitivity of R_{aux} to it was computed. A gradient’s size is a proxy: Section 4 shows that the decline is in the auxiliary gradient itself, not what that gradient would have done. The momentum dependence is descriptive, and the pre-set proportional relationship was not met.

Conclusion. Auxiliary objectives differ in how long they keep training the encoder, and final return cannot show it. In this study SPR’s auxiliary gradient declined steadily through every run while CURL’s and reconstruction’s held, the fall was in the auxiliary gradient itself, and it followed the early saturation of SPR’s moving-average target. Logging the auxiliary gradient costs two backward passes. A result from an objective whose gradient faded says little about the idea behind it.

References

- Michael Beukman, Khimya Khetarpal, Zeyu Zheng, Will Dabney, Jakob Foerster, Michael Dennis, and Clare Lyle. Preventing learning stagnation in PPO by scaling to 1 million parallel environments. *arXiv preprint arXiv:2603.06009*, 2026.
- Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 794–803, 2018.
- Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 2048–2056, 2020.
- Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632:768–774, 2024.
- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1407–1416, 2018. arXiv:1802.01561.
- Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.
- Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, Michal Valko, Rémi Munos, Mohammad Gheshlaghi Azar, and Bilal Piot. BYOL-Explore: Exploration by bootstrapped prediction. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

- Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017.
- Arthur Juliani and Jordan T. Ash. A study of plasticity loss in on-policy deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.
- Abhisek Keshari. What does a ProcGen generalization gap measure? Action rules, residual entropy, and the missing random floor. arXiv preprint arXiv:2609.32532v2, 2026.
- Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5639–5650, 2020.
- Xiang Li, Jinghuan Shang, Srijan Das, and Michael S Ryoo. Does self-supervised learning really improve reinforcement learning from pixels? In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022.
- Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan Pascanu, and Caglar Gulcehre. No representation, no trust: Connecting representation, collapse, and trust issues in PPO. In *Advances in Neural Information Processing Systems*, volume 37, 2024.
- Tianwei Ni, Benjamin Eysenbach, Erfan SeyedSalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive RL. In *International Conference on Learning Representations*, 2024.
- Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.
- Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.
- Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.
- Yunhao Tang, Zhaohan Daniel Guo, Pierre Harvey Richemond, Bernardo Avila Pires, Yash Chandak, Rémi Munos, Mark Rowland, Mohammad Gheshlaghi Azar, Charline Le Lan, Clare Lyle, András György, Shantanu Thakoor, Will Dabney, Bilal Piot, Daniele Calandriello, and Michal Valko. Understanding self-predictive learning for reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 33632–33656, 2023.
- Claas A. Voelcker, Tyler Kastner, Igor Gilitschenski, and Amir-massoud Farahmand. When does self-prediction help? Understanding auxiliary tasks in reinforcement learning. *Reinforcement Learning Journal*, 2024.
- Tao Yu, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. Mask-based latent reconstruction for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

The full technical report, with the defect log and registration chain, is at <https://github.com/AbhisekKeshari/aux-loss-liveness>.

A Configuration

Every setting is identical across conditions except the auxiliary objective. Table 4 is read from the configuration each run stored in its own checkpoints, not from the repository’s configuration file. The baseline, SPR, CURL, reconstruction and shuffled-CURL runs, 45 in all, agree on all 44 fields that affect the computation, apart from the game, the seed and the auxiliary objective. Every run uses 16 parallel environments, with no reward normalization or clipping. SPR predicts 3 steps ahead, as every run’s stored configuration records; the configuration file’s current default is 5. The momentum-sweep runs differ from the core SPR runs on miner only in τ , run length and logging cadence, which a configuration diff checked before each launch.

Table 4: Training configuration, as stored in every run’s checkpoints.

PPO		Encoder and auxiliary objectives	
Games	coinrun, heist, miner	Encoder	IMPALA CNN
Distribution mode	easy	Feature width	256
Training levels	200	CURL coefficient	0.1
Environment steps	8M	CURL crop size	60
Parallel environments	16	CURL temperature	0.1
Rollout length	256	SPR coefficient	2.0
Minibatches	8	SPR prediction horizon	3
Update epochs	3	SPR target momentum τ	0.99
Learning rate (constant)	0.0005	SPR target update	once per PPO update
Discount γ	0.999	SPR projection width	256
GAE λ	0.95	Reconstruction coefficient	0.5
Clip coefficient	0.2		
Entropy coefficient	0.01		
Value coefficient	0.5		
Max gradient norm	0.5		
Advantage normalization	yes		
Value-loss clipping	yes		

B Per-run shelf-life results

Table 5 gives every run’s terminal-over-early ratio of the auxiliary gradient norm (R_{aux} , the primary statistic), of the auxiliary norm over the PPO norm ($R_{A/P}$), and of the share (R_{share}), with the early step at 503,808 for every run. On R_{aux} the groups overlap at one run: SPR’s highest ratio is 0.121 and CURL’s lowest is 0.100. On $R_{A/P}$ 2 CURL or reconstruction runs sit at or below SPR’s highest ratio; on the share none does. Shuffled CURL is a control and is discussed in Appendix E; one of its runs has a zero auxiliary gradient at an anchor, so its ratios are undefined.

C Re-evaluation under the companion protocol

Every final checkpoint was scored with the companion paper’s harness (Keshari, 2026): both action rules, sampled primary; three seeded level draws on the training levels and on the held-out levels; the uniform-random floor on the same draws; and merged-action entropy with the companion paper’s per-game equivalence classes. Each run’s value is its mean over the three draws. The floors reproduce the companion paper’s draw by draw (Table 6). The statistical plan is in Appendix G.

Table 7 gives every contrast with the baseline. Holm adjustment runs within the family of 9 SPR, CURL and reconstruction contrasts, per metric; shuffled CURL is reported beside them and stays outside the family. “Margin” is the smallest equivalence margin, in return units, at which each contrast would be declared

Table 5: Shelf-life ratios, every run.

Objective	Game	Seed	R_{aux}	$R_{A/P}$	R_{share}
SPR	coinrun	1	0.011	0.015	0.016
SPR	coinrun	2	0.039	0.080	0.082
SPR	coinrun	3	0.043	0.095	0.097
SPR	heist	1	0.121	0.052	0.077
SPR	heist	2	0.092	0.035	0.059
SPR	heist	3	0.045	0.021	0.029
SPR	miner	1	0.009	0.002	0.003
SPR	miner	2	0.015	0.002	0.005
SPR	miner	3	0.010	0.002	0.003
CURL	coinrun	1	0.556	0.620	0.712
CURL	coinrun	2	0.602	0.893	0.903
CURL	coinrun	3	1.501	0.351	0.454
CURL	heist	1	0.389	0.104	0.199
CURL	heist	2	0.100	0.032	0.149
CURL	heist	3	2.098	0.844	0.894
CURL	miner	1	0.741	0.137	0.238
CURL	miner	2	0.248	0.049	0.154
CURL	miner	3	1.844	0.497	0.571
reconstruction	coinrun	1	0.613	0.640	0.642
reconstruction	coinrun	2	0.697	0.910	0.910
reconstruction	coinrun	3	1.331	0.694	0.696
reconstruction	heist	1	0.832	0.151	0.156
reconstruction	heist	2	0.258	0.203	0.213
reconstruction	heist	3	0.794	0.677	0.684
reconstruction	miner	1	0.835	0.590	0.601
reconstruction	miner	2	0.578	0.132	0.143
reconstruction	miner	3	1.270	0.283	0.300
shuffled CURL	coinrun	1	0.609	0.457	0.462
shuffled CURL	coinrun	2	—	—	—
shuffled CURL	coinrun	3	0.029	0.015	0.016
shuffled CURL	heist	1	0.921	0.610	0.623
shuffled CURL	heist	2	0.403	0.194	0.206
shuffled CURL	heist	3	1.632	0.431	0.453
shuffled CURL	miner	1	0.976	0.389	0.398
shuffled CURL	miner	2	1.197	0.244	0.257
shuffled CURL	miner	3	3.103	0.500	0.521

equivalent; the test’s own margin follows the companion paper’s rule (Appendix G). “vs. floor” is the arm’s call against the held-out floor (n.d.: not distinguishable). Table 8 gives every run.

Table 6: Uniform-random floors, mean over three seeded draws.

Game	Training levels	Held-out levels
coinrun	2.94	3.28
heist	3.54	3.15
miner	1.24	1.17

D Robustness of the measurement

These checks were run after the analysis in Section 3, and none of them changes a number reported there.

Table 7: Every contrast with the baseline, three tested metrics.

Game	Arm	n	Sampled held-out return (primary)					Sampled training		Merged entropy		vs. floor
			diff.	p	Holm	TOST p	margin	diff.	Holm	diff.	Holm	
coinrun	SPR	3/3	+0.26	0.389	1.000	0.403	0.92	+0.35	1.000	-0.045	1.000	above
coinrun	CURL	3/3	+0.43	0.077	0.563	0.724	0.82	+0.80	0.289	-0.006	1.000	above
coinrun	reconstruction	3/3	+0.00	1.000	1.000	0.041	0.30	+0.33	1.000	-0.020	1.000	above
coinrun	shuffled CURL	3/3	+0.28	0.602	-	0.461	1.58	+0.31	-	+0.012	-	above
coinrun	baseline											above
heist	SPR	3/3	-0.04	0.781	1.000	0.091	0.44	+0.12	1.000	+0.038	1.000	below
heist	CURL	3/3	+0.74	0.070	0.563	0.910	1.34	+0.91	0.345	-0.054	1.000	n.d.
heist	reconstruction	3/3	-0.43	0.261	1.000	0.637	1.22	-0.16	1.000	+0.006	1.000	below
heist	shuffled CURL	3/3	+0.32	0.249	-	0.511	0.90	+0.00	-	+0.109	-	below
heist	baseline											below
miner	SPR	3/6	-0.74	0.390	1.000	0.720	2.32	-1.25	1.000	+0.088	0.247	n.d.
miner	CURL	6/6	+1.93	0.017	0.156	0.985	3.05	+2.52	0.145	-0.055	1.000	above
miner	reconstruction	3/6	-1.16	0.093	0.563	0.917	2.29	-1.03	1.000	+0.102	0.154	above
miner	shuffled CURL	3/6	+0.30	0.673	-	0.530	1.61	+0.14	-	+0.088	-	above
miner	baseline											above

Minibatch noise. A single logged share is one minibatch’s draw. We reloaded 18 checkpoints, covering every objective and game at two points in training. At each we collected three rollouts and computed the share on every minibatch, 24 draws per checkpoint, with no optimizer step in between. The coefficient of variation ranges from 0.18 to 0.55 (median 0.29). Each terminal-over-early ratio in Section 3 compares two such draws. SPR’s median R_{aux} of 0.039 lies far outside this noise; differences of tens of percent between individual CURL and reconstruction runs do not.

Numerator and denominator, by game. Section 4 gives the pooled split. By game, the medians of f_{num} are 1.29 on coinrun, 0.72 on heist and 0.74 on miner. SPR’s auxiliary gradient falls by a median factor of 26 (range 8–116). The trend of PPO’s encoder gradient follows the game in every arm: its terminal-over-early ratio, over all auxiliary arms in a game, has a median of 0.93 on coinrun, 2.41 on heist and 4.82 on miner, and within a game the arms share the direction and order of magnitude of that ratio (Figure 5). Its level does not: CURL’s P sits above the other arms’ on heist and miner, a difference this study does not test and the decomposition does not use. Figure 4 shows every auxiliary gradient trajectory at its native cadence, including the controls.

E Controls

Neither CURL control isolates contrastive structure, because each changed the size of the auxiliary gradient as well. A control meant to remove structure has to hold gradient size fixed, and ours did not.

Shuffled CURL. Shuffled CURL keeps CURL’s machinery but scrambles which augmented view is the positive, so the contrastive task cannot be solved. It also shrank the auxiliary gradient. Its mean terminal auxiliary gradient sits 5.7 to 16.2 times below the smallest value any CURL run in the same game logged at any step. A comparison between shuffled CURL and CURL therefore changes structure and gradient size at once, and cannot attribute a difference to either.

Its auxiliary gradient is also erratic. On coinrun it is exactly zero at several logged steps (marked on the bottom axis of Figure 4). On coinrun seed 2 it is exactly zero at the terminal measurement. The plan excludes a run with a zero auxiliary gradient at either anchor from R_{aux} , so that run has no ratio. The other runs’ ratios are listed in Appendix B; shuffled CURL is left out of Figure 2, Table 1 and every range in Section 3. Over its 8 defined runs, shuffled CURL’s R_{aux} has a median of 0.95 (range 0.03–3.10).

CURL at matched low gradient. `curl_weak` keeps CURL’s real positives and scales its coefficient down per game, so that its auxiliary gradient matches shuffled CURL’s. We derived the multipliers from the linearity of the gradient in the coefficient, and set a band of a factor of two either way before launch. The realised gradient landed at $\times 0.84$ to $\times 2.05$ of target: $\times 0.844$ on heist, $\times 1.987$ on miner and $\times 2.046$ on coinrun, outside the band. Coinrun’s seeds agree with each other to within 3%, so its miss is a systematic

Table 8: Every checkpoint under the companion protocol.

Game	Arm	Seed	Sampled return		Greedy return		Merged entropy
			held-out	training	held-out	training	
coinrun	baseline	1	6.35	6.22	3.67	4.19	1.37
coinrun	baseline	2	6.56	6.35	5.29	5.18	1.42
coinrun	baseline	3	6.41	6.20	5.13	5.00	1.41
coinrun	SPR	1	7.06	6.77	4.53	4.69	1.31
coinrun	SPR	2	6.80	7.01	3.18	2.81	1.31
coinrun	SPR	3	6.25	6.04	4.74	4.58	1.45
coinrun	CURL	1	6.85	6.82	5.78	5.31	1.41
coinrun	CURL	2	7.14	7.40	5.47	5.26	1.35
coinrun	CURL	3	6.64	6.95	5.42	5.73	1.42
coinrun	reconstruction	1	6.64	6.95	4.74	5.16	1.34
coinrun	reconstruction	2	6.25	6.64	5.03	5.31	1.39
coinrun	reconstruction	3	6.43	6.17	4.61	4.66	1.41
coinrun	shuffled CURL	1	7.60	7.81	5.16	5.86	1.30
coinrun	shuffled CURL	2	6.43	6.12	4.40	4.90	1.42
coinrun	shuffled CURL	3	6.12	5.78	4.61	4.92	1.52
heist	baseline	1	1.88	4.11	0.29	0.99	1.67
heist	baseline	2	1.93	4.30	0.00	1.07	1.72
heist	baseline	3	1.93	4.30	0.23	0.86	1.80
heist	SPR	1	1.77	4.66	0.00	1.02	1.79
heist	SPR	2	2.14	4.01	0.29	1.90	1.81
heist	SPR	3	1.69	4.40	0.16	1.69	1.71
heist	CURL	1	2.55	5.08	0.26	2.21	1.67
heist	CURL	2	3.05	5.57	0.68	2.99	1.71
heist	CURL	3	2.34	4.79	0.52	1.51	1.65
heist	reconstruction	1	0.94	4.43	0.00	1.30	1.67
heist	reconstruction	2	1.72	4.01	0.18	1.46	1.79
heist	reconstruction	3	1.80	3.80	0.21	1.38	1.75
heist	shuffled CURL	1	2.53	4.45	0.00	1.02	1.90
heist	shuffled CURL	2	2.32	4.19	0.42	0.81	1.80
heist	shuffled CURL	3	1.85	4.06	0.00	0.99	1.82
miner	baseline	1	1.71	2.33	0.13	0.18	1.23
miner	baseline	2	5.09	6.98	0.42	1.28	1.14
miner	baseline	3	5.14	6.34	0.48	1.02	1.22
miner	baseline	4	4.72	6.83	0.44	1.27	1.16
miner	baseline	5	4.74	5.97	0.47	1.29	1.10
miner	baseline	6	5.33	6.40	0.55	0.91	1.03
miner	SPR	1	3.48	4.36	0.22	0.51	1.24
miner	SPR	2	4.80	5.83	0.37	0.61	1.22
miner	SPR	3	2.85	3.48	0.12	0.36	1.24
miner	CURL	1	6.07	7.88	0.82	1.65	1.19
miner	CURL	2	6.33	8.42	0.90	1.78	1.02
miner	CURL	3	6.58	8.39	0.97	1.72	1.02
miner	CURL	4	6.17	8.17	0.70	1.48	1.17
miner	CURL	5	6.40	8.57	0.74	2.10	1.04
miner	CURL	6	6.73	8.51	0.84	1.76	1.10
miner	reconstruction	1	3.49	5.30	0.39	0.78	1.23
miner	reconstruction	2	3.28	4.82	0.46	0.60	1.26
miner	reconstruction	3	3.12	4.20	0.23	0.47	1.25
miner	shuffled CURL	1	4.01	4.62	0.33	0.55	1.29
miner	shuffled CURL	2	4.88	6.61	0.55	0.87	1.20
miner	shuffled CURL	3	5.39	6.61	0.51	1.10	1.22

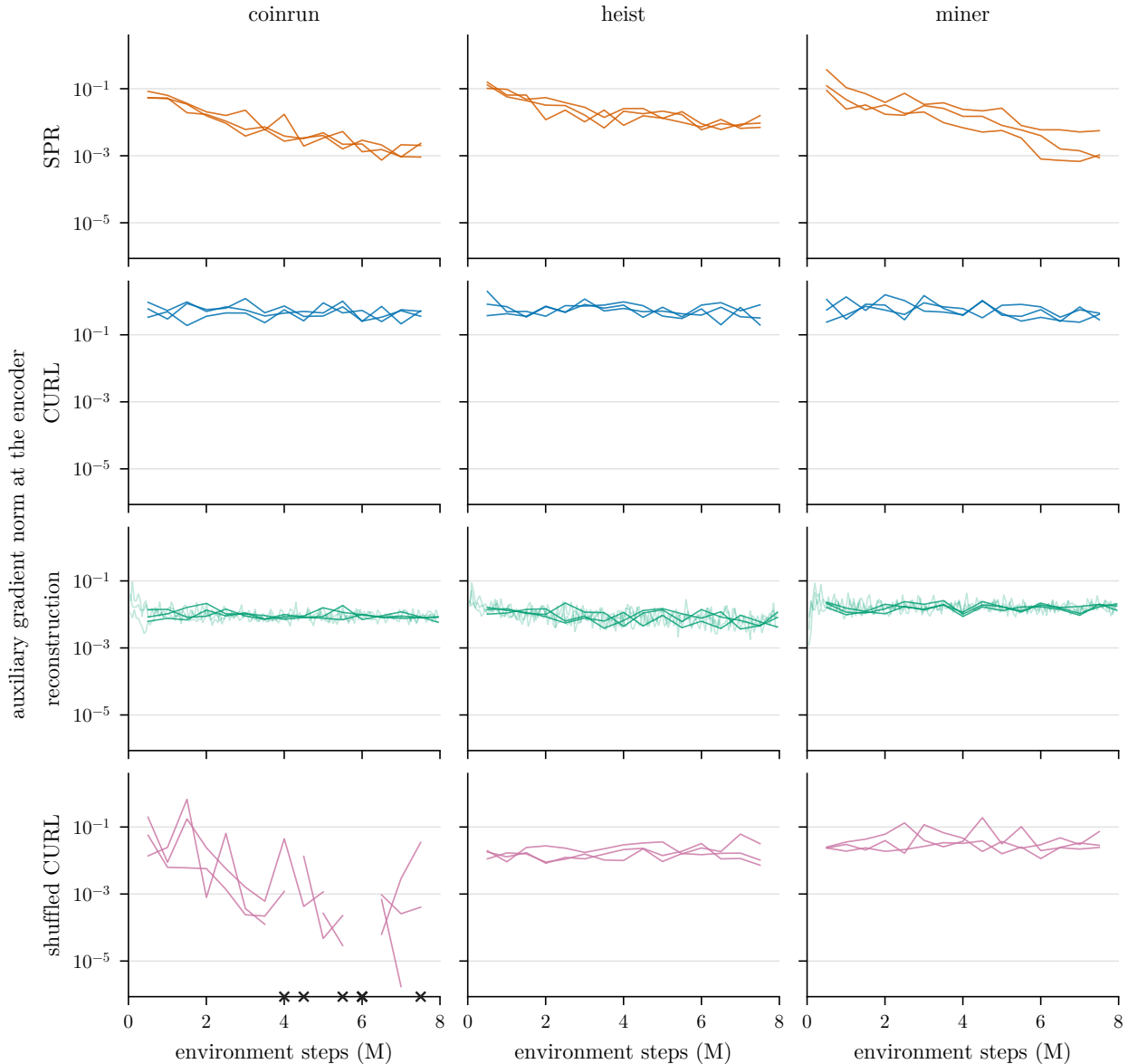


Figure 4: Auxiliary gradient norm at the encoder, every run drawn unsmoothed, rows by objective and columns by game, shared log scale. Faint: every logged step; solid: the common 500K grid used in Figure 1. A step at which the logged norm is exactly zero (shuffled CURL on coinrun) breaks the line and is marked $\times$ on the bottom axis.

offset rather than noise. The derivation holds for one gradient at fixed parameters, and did not hold over training. A plausible reason, which we did not test, is that the coefficient changes the trajectory along which the gradient is measured. Anyone matching gradient size should calibrate the multiplier on a short pilot run. The arm ran 3, 3 and 6 seeds on coinrun, heist and miner. It is not part of the re-evaluation in Appendix C, and we draw no conclusion about structure from it.

Augmentation only. Naive augmentation inside PPO diverged in 5 of 5 runs, as DrAC predicts (Raileanu et al., 2021). This arm applied CURL’s random crops inside the PPO update with no auxiliary loss (2 runs on coinrun, 2 on heist, 1 on miner). Minibatch observations were cropped for the forward pass, while the stored log-probabilities, advantages and returns came from uncropped rollouts. The importance ratio then

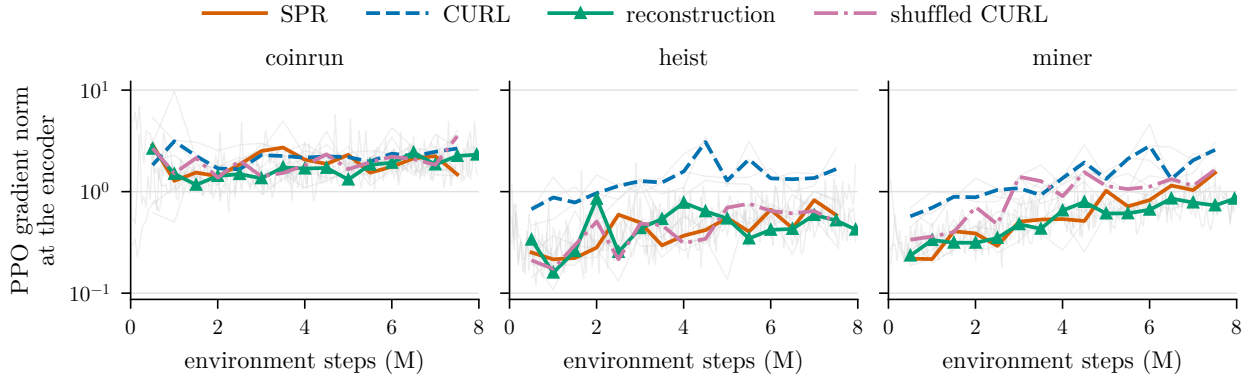


Figure 5: PPO’s encoder gradient norm P over training: the median over seeds per objective on the common 500K grid in colour, every run at its native cadence in grey behind, log scale. The trend of P follows the game in every arm; CURL’s level is higher on heist and miner, which is untested here and not used by the decomposition.

measures crop noise rather than a change in the policy, and the value head regresses targets for observations it no longer sees. The value loss diverged, and policy entropy reached exactly zero in 5 of 5 runs. The arm carries no results.

F The liveness probe

Section 7 describes the probe; Table 9 gives it at three lengths. A probe trains to its window and asks whether SPR’s own cosine to its target has reached 0.99. On the laptop GPU this study used, a 500K-step probe costs 9.4 minutes. Charging it to every run and crediting only the flagged ones, it would have saved 15.2 of the 67.3 hours the sweep cost (22.6%). The best length is the one that just clears saturation: shorter probes catch nothing, longer ones pay more on every run that turns out to be alive. Run inside the training job rather than as a separate run, the probe costs only its logging. The analysis is retrospective, on the 9 core SPR runs.

Table 9: The liveness probe at three lengths: cost per run, SPR runs flagged, and net saving against the 67.3 hours of the core sweep.

Probe length	Cost per run	SPR runs flagged	Net saving
250K	4.7 min	0 of 9	−1.6 h
500K	9.4 min	8 of 9	+15.2 h
1M	18.9 min	9 of 9	+14.5 h

G Statistical procedures

This appendix states the decision rules behind Sections 3, 5 and 6, and how each came out.

Shelf life on the auxiliary gradient. The rule for the primary statistic was not met, and the text reports overlap instead. For each run, $R_{\text{aux}} = A_{\text{term}}/A_{\text{early}}$. Early is the logged step nearest 500K within 250K, which is step 503,808 in every run; terminal is the last logged step. Two secondary statistics use the same rows: the ratio of A/P and the ratio of the share. The population is the 9 runs each of SPR, CURL and reconstruction. Shuffled CURL is reported separately (Appendix E). A run with a zero auxiliary gradient at either anchor is excluded from a statistic, with its reason.

The plan allowed a separation claim only if SPR’s largest R_{aux} lay below the smallest CURL or reconstruction value. Otherwise the text reports medians, ranges and the number of other runs at or below SPR’s largest. One CURL run, at 0.100, lies below SPR’s largest, 0.121, so 1 run overlaps and the claim is not made. On the share the ranges do not overlap: SPR’s largest ratio is 0.097 and the smallest other is 0.143. The plan requires a disagreement between the primary and a secondary statistic to be reported without choosing between them, which the caption of Table 1 does.

Momentum sweep. The rule was not met: the slope fell outside its window. SPR’s target is saturated at the first update after which the cosine between prediction and target stays at or above 0.99 to the end of the run. The sweep logged the cosine at every update. “To the end” is evaluated over each run’s own length, so the definition depends on run length. A cosine that dips below the threshold late counts as saturated in a short run and not in a longer one. Run length differs across τ , so this matters when comparing values of τ .

The rule was an ordinary least-squares fit of log saturation update on $\log 1/(1 - \tau)$, one point per run. A slope in $[0.7, 1.3]$ with R^2 above 0.9, and at most one value of τ censored, would have licensed a scaling claim. A run is censored if its cosine is below the threshold at its last logged update. The fit over all 15 runs, none censored, gave slope 0.63 and $R^2 = 0.95$. Section 5 therefore reports the sweep as found. The core runs at $\tau = 0.99$ logged the cosine only every 50 updates and do not enter the fit; 1M-step re-runs at the same τ take their place. The horizon study in Table 2 is a separate, shorter study: 3 runs on miner, one seed per horizon, of 1M steps each, with the cosine logged every 3–10 updates. Its saturation steps are on a different grid from the core runs’ and are not interchangeable with them.

Post-saturation check. The plan’s secondary, descriptive check asks whether the share fell below a quarter of its first post-saturation value before the run ended. It did in all 9 runs at $\tau = 0.90$ – 0.98 and in 2 of 3 at $\tau = 0.995$, and in none of the 1M-step runs at $\tau = 0.99$ within their length. The check starts at saturation, so it says nothing about the order of saturation and collapse, and no claim rests on it.

Re-evaluation. Under Holm’s correction, no contrast survives on any tested metric. The unit of analysis is the run, scored as its mean over three seeded level draws. Each auxiliary arm is compared with the baseline in the same game by Welch’s two-sided t -test. Holm’s correction runs within the family of 9 contrasts (SPR, CURL and reconstruction in each game), separately for each metric. Sampled held-out return is primary. Sampled training return and merged-action entropy are secondary. Greedy returns are reported but not tested. Shuffled CURL’s contrasts and the check on miner seeds 4–6 are reported outside the family. Arms have 3 runs, except miner’s baseline and CURL, which have 6.

Equivalence uses two one-sided tests with the companion paper’s margin, $\delta = \max(0.1 \lfloor \text{floor} \rfloor, 0.25)$ return units, where the floor is the uniform-random return on the same level set (Keshari, 2026). That paper applies the margin to an arm against the floor. We apply it to an arm against the baseline, with the Welch standard error and degrees of freedom. The margin depends only on the floor, so it was fixed before any policy was scored. We report the raw TOST p -value, as the companion paper does, with the Holm-adjusted value beside it. For each return contrast we also report the smallest margin at which equivalence would be declared (Appendix C). Merged-action entropy is not tested for equivalence.

Every forward pass ran with TF32 disabled. With TF32 on, logits computed through the companion harness and through this study’s own pipeline differed at the level of GPU rounding noise. Disabling it puts every arm on the same numeric path.

Registration. The analysis plan was committed before any statistic it scores was computed. It was changed only by a dated amendment file, also committed before the numbers it affects. The one amendment concerns the numerical precision of the re-evaluation and disabled TF32. The full technical report also uses an extinction threshold on the share; it was set after the SPR and CURL runs and before the reconstruction runs, and this paper does not use it. The probe analysis of Section 7 is retrospective. The plan, the amendment and their hashes are in the supplementary files.

H Measurement checks

Hook placement and gradient clipping. Measuring the share before gradient clipping gives the same value as measuring it after. The update clips by one global factor over all parameters, and a factor common to A and P cancels in the share. A standalone probe checks this on archived checkpoints. It imports the training code and nothing from the analysis pipeline, rebuilds the agent and auxiliary modules, collects a fresh rollout and draws one minibatch. It then computes the share four ways from one forward pass (Table 10). It ran on 9 checkpoints, one per game and objective, at 4M steps.

The clip was active in 7 of the 9 cells, with factors down to 0.20. The share under the global clip matches the logged share to within $2.2\text{e-}16$ in every cell. Clipping each loss’s gradient separately would not cancel, and would raise the share by up to 4.4 times. Dividing the coefficient back out changes the share by factors from 0.50 to 4.6, in both directions, so the paper reports the share after the coefficient is applied. R_{aux} does not depend on this choice, since a constant coefficient cancels from it.

Table 10: The auxiliary share on one minibatch, computed four ways: as logged, with the coefficient divided out, under the global clip the update applies, and under a per-loss clip the update does not apply. “factor” is the global clip factor; a factor of one means no clipping occurred.

env	objective	clipped	factor	auxiliary share of the encoder gradient			
				pipeline	pre-coef.	clip (joint)	clip (per-loss)
coinrun	CURL	yes	0.222	0.15698	0.65062	0.15698	0.44399
coinrun	SPR	yes	0.195	0.00274	0.00137	0.00274	0.01200
coinrun	reconstruction	yes	0.223	0.00763	0.01515	0.00763	0.02979
heist	CURL	yes	0.298	0.41154	0.87490	0.41154	0.50000
heist	SPR	no	1.000	0.04250	0.02171	0.04250	0.04250
heist	reconstruction	no	1.000	0.03650	0.07044	0.03650	0.03650
miner	CURL	yes	0.336	0.13217	0.60364	0.13217	0.31899
miner	SPR	yes	0.790	0.00899	0.00452	0.00899	0.01009
miner	reconstruction	yes	0.351	0.01265	0.02498	0.01265	0.03422

Bit-identity. Turning the hook on leaves training unchanged. Two seed-matched SPR runs on coinrun, 1M steps each, differed only in whether the hook was enabled. The instrumented run measured the share 20 times, a denser cadence than the core runs used. All 934 tensors across 4 matched checkpoints are byte-identical. All 21 logged loss columns that the hook does not write agree at every shared step. The core runs’ own cadence was not compared directly, but it runs the same hook code.

The momentum sweep logged more often than the core runs, and this did not change training either. Two short SPR runs on miner that differed only in how often losses are written produced identical checkpoints and identical logged losses. Each $\tau = 0.99$ re-run, with the sweep’s denser logging, matches its core run’s checkpoint at step 503,808 in every tensor, and its logged losses at every step both runs recorded.