FROM RANKING TO SELECTION: EVALUATING REPRESENTATION PROXIES IN RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretraining representations can improve sample efficiency in reinforcement learning, but evaluating a representation normally requires expensive finetuning across many seeds and tasks. To avoid this cost, prior work scores representations with cheap proxies, such as linear probing against task-derived targets (e.g. reward) and direct measures of the representation itself (e.g. effective rank), and validates them by their *rank correlation* (ρ) with downstream performance. Such validation is incomplete, as ρ scores the ordering of an entire pool of candidates, whereas the use case that matters in practice, model selection, depends only on the top choices. We therefore propose *normalized fraction-of-oracle* (NFoO), which scores a proxy by the downstream performance of the representation it selects relative to the best available, and argue it should be reported alongside ρ. We also widen the pool of proxies under test: we evaluate a broad range of probing targets, spanning reward presence, multi-step return, behavioral cloning, episode structure, transition dynamics, and temporal structure. We assess seven proxies under both metrics in two model selection scenarios on the Atari benchmark: selecting among pretraining methods and among representations within a single game. We also evaluate their use for early stopping during pretraining. Across more than 20,000 finetuning runs on 15 games, we find an asymmetry: across proxies that capture different signals, proxy rankings shift with the candidate pool and downstream learner, yet their selections remain consistently strong. Representation proxies are therefore more robust as selectors than their rank correlations alone would suggest.

1 INTRODUCTION

Pretraining representations has become a standard technique across domains including Computer Vision (Grill et al., 2020), Natural Language Processing (Devlin et al., 2019), and Reinforcement Learning (RL), as it can yield general-purpose features that improve downstream sample efficiency and transfer. In RL specifically, evaluating a pretrained representation requires online finetuning, which is expensive to do reliably across many seeds, tasks, methods, and checkpoints (Henderson et al., 2019; Agarwal et al., 2021). As a result, a growing body of work has proposed cheap proxies which are computable from a frozen representation alone, that aim to substitute for this expensive evaluation (Zhang et al., 2024; Anand et al., 2019; Dittadi et al., 2022).

Existing proxies mostly fall into two kinds. The first probes a frozen representation for decodable structure: Anand et al. (2019) use ground-truth states of Atari games, Zhang et al. (2024) use rewards and actions, and Fu et al. (2022) probe offline RL agents for transition dynamics and optimal-policy quantities. The second measures a property of the representation directly, such as effective rank (Kumar et al., 2021), which measures representation collapse. When these proxies are evaluated as predictors of downstream performance, prior work has relied almost exclusively on rank correlation (Spearman ρ) (Zhang et al., 2024; Lee et al., 2023). Rank correlation scores the ordering of an entire pool, whereas model selection, the use case that matters in practice, depends only on the top choices: a low ρ may hide correct identification of the best representation, and a high ρ is compatible with missing it. Moreover, no prior work has evaluated a broad set of proxies together, leaving it unclear how they compare against one another.

In this work, we assess seven proxies (reward, n-step return, action, inverse action, terminal, temporal distance, effective rank) on both ranking and selection, reporting ρ alongside normalized

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fraction-of-oracle (NFoO), a new metric measuring the normalized ratio between the downstream performance of the selected representation and that of the best in hindsight. We evaluate all proxies in a large-scale study spanning 15 Atari games, 16 pretraining methods, and 2 downstream learners, organized as a sequence of selection problems: choosing among pretraining methods, repeating that choice with a second learner, choosing among 100 candidates within a single game, and deciding when to stop a single pretraining run.

Across these settings, we find an asymmetry: a proxy's rank correlation can suggest it is unreliable, yet its selections are consistently strong. The reward probe achieves ρ even as low as 0.6 and still selects the best available method (see Section 5.2). Moreover, we find that every proxy we test selects well. When choosing among pretraining methods, random selection attains an NFoO of only 0.39 (across both Rainbow and PPO), while most proxies reach 0.8 with their top pick and over 0.9 with their top two. The same holds within a single game: every proxy beats random selection in at least 12 of the 15 games. Selection quality is likewise stable across downstream learners: the two learners order the 16 methods only moderately alike ($\rho = 0.62$), yet proxies identify near-oracle methods under both, even though rank correlation does not transfer between them. Proxies can also guide the early stopping of pretraining. Since later checkpoints often perform similarly, continuing to train past this plateau wastes compute. With a simple rule that halts once proxy scores stop improving, every proxy outperforms stopping at a fixed checkpoint at comparable compute and typically stops earlier than the same rule applied to validation loss.

From rank correlation (ρ) to model selection (NFoO), proxies are considerably more robust as selectors than their rank correlations alone would suggest. The metrics, though, are not interchangeable: ρ can misjudge how well a proxy selects, while NFoO reflects only the pool at hand and can hide biases that ρ exposes. We recommend reporting both. In summary, our contributions are:

- **A selection-aligned metric.** We introduce *normalized fraction-of-oracle*, which explicitly measures the performance of proxies for model selection. We report it alongside ρ and highlight instances where the two metrics disagree.
- **Which proxy to choose matters little for selection.** We assemble seven proxies spanning reward presence, multi-step return, behavioral cloning, episode structure, transition dynamics, temporal structure, and representation collapse, and find that selection quality varies little with the choice of proxy even though they rely on different signals.
- **A large-scale evaluation.** We finetune over 20,000 runs across 16 pretraining methods, 15 Atari games, and 2 downstream learners, finding that every proxy selects well above chance and guides an early stopping rule to halt pretraining early.

2 Related Work

Representation pretraining in RL. A substantial body of work pretrains state representations prior to RL, primarily motivated by the promises of more sample-efficient downstream learning (Schwarzer et al., 2021b) and transfer across tasks (Taiga et al., 2023; Kim et al., 2024). Pretraining objectives span the major self-supervised learning paradigms: contrastive methods (Stooke et al., 2021; Anand et al., 2019; Srinivas et al., 2020), self-predictive methods (Schwarzer et al., 2021a;b), and generative methods (Kingma & Welling, 2022). This work focuses more so on sample efficiency than transfer; we follow Schwarzer et al. (2021b) in that we pretrain on data from the same task used for downstream finetuning, which we evaluate primarily in the Atari100k benchmark (Łukasz Kaiser et al., 2020), and following Stooke et al. (2021), we freeze the pretrained encoder during finetuning, decoupling representation learning from RL.

Evaluating representation quality in RL. The most straightforward way to evaluate a representation for RL is to actually perform online finetuning on target downstream tasks (Stooke et al., 2021). However, doing so can be very costly as many seeds are typically required in RL (Agarwal et al., 2021; Henderson et al., 2019); thus over many models and many tasks, this evaluation scheme becomes prohibitive. A natural proxy is to use the pretraining loss itself, but losses from different objectives are not on a common scale and thus cannot be meaningfully compared across pretraining methods (Stooke et al., 2021). Other approaches fall broadly into two threads: probing frozen representations for decodable structure, and measuring properties of the representation directly.

Linear probing (Alain & Bengio, 2018), which originates in computer vision and natural language processing (Conneau et al., 2018; Tenney et al., 2019), fits a linear map from frozen representations to targets of interest, treating decodability as a proxy for representation quality. In RL, Anand et al. (2019) probe for ground truth state variables extracted from a suite of Atari games and find that different self-supervised objectives capture different subsets of them. Most directly related to our work, Zhang et al. (2024) evaluate reward and action probing as predictors of downstream finetuning performance. We build on their setup with a larger set of proxies, games, and pretraining methods, but depart from it methodologically: rather than relying on rank correlation alone, we evaluate proxies directly against the model-selection use case.

A second thread measures properties of the representation directly. Kumar et al. (2021) identify rank collapse in value-based RL and relate effective rank to downstream performance. Gulcehre et al. (2022) complicate this picture, showing that the association largely disappears under more extensive hyperparameter sweeps. Similar work characterizes the phenomenon more broadly as a loss of capacity or plasticity over training (Lyle et al., 2022; 2023), including through dormant units (Sokar et al., 2023). We include effective rank among our proxies, noting that these works study how representations evolve while trained jointly with an RL objective, whereas we decouple pretraining and RL and hold the representation frozen.

Model selection with proxies. The problem of ranking candidate models by a cheap proxy, rather than by the expensive procedure of interest, recurs across machine learning. In neural architecture search, cheap proxies score architectures from a single minibatch at initialization (Mellor et al., 2021; Abdelfattah et al., 2021). Closest to our setting, offline policy selection asks which of several trained policies to deploy without online evaluation (Paine et al., 2020). This literature, in particular the DOPE benchmark (Fu et al., 2021), evaluates algorithms via both rank and model selection metrics. We adopt this perspective of using multiple metrics to evaluate representation selection in RL (see Section 3), where prior work has so far evaluated proxies with rank correlation alone.

3 Methodology

3.1 Proxies

An offline dataset contains many potentially informative signals for measuring representation quality, and we evaluate seven proxies drawing on different ones. Each is a linear probe fitted to a target derived from the offline data, with the exception of effective rank, which is computed from the representation directly. Binary classification is scored with F_1, multi-class classification with weighted F_1, and regression with R^2. Below, we denote by $\varphi(s_t)$ the representation of state s_t.

Reward probing (Zhang et al., 2024). Given $\varphi(s_t)$, predict reward r_t, implemented as binary classification with $\mathbf{1}_{r_t \neq 0}$ as the targets. This asks whether the representation encodes *reward presence* in the current state. Reward prediction was used as an auxiliary loss in Jaderberg et al. (2016).

n-step return probing. Given $\varphi(s_t)$, predict $\sum_{h=0}^{n-1} \gamma^h r_{t+h}$, implemented as regression. This extends reward probing across time, asking whether the representation supports estimating *multi-step value*. We use the same value of γ as in the downstream learner, and use $n = 10$ in our main experiments; we ablate the choice of n in Appendix G.4.

Action probing (Zhang et al., 2024). Given $\varphi(s_t)$, predict the action a_t taken by the dataset's behavior policy, implemented as multi-class classification. This asks whether the representation supports *behavior cloning* of the dataset's policy.

Inverse action probing (Fu et al., 2022). Given $\varphi(s_t)$ and $\varphi(s_{t+1})$, predict the action a_t, implemented as multi-class classification. Unlike action probing, the probe observes the outcome of the action, so the target is determined by the transition itself rather than by the policy that produced it; this measures whether the representation captures the environment's *transition structure*. It has been used widely as an auxiliary loss (Shelhamer et al., 2017; Pathak et al., 2017; Agrawal et al., 2016).

Terminal probing. Given $\varphi(s_t)$, predict termination $d_t \in \{0, 1\}$ indicating whether s_t ends the episode, implemented as binary classification; the same signal is used as an auxiliary loss by Leibfried & Vrancx (2018). Termination is the coarsest form of *episode structure*, and predicting it

requires the representation to encode proximity to failure or completion. The target is sparse, with one positive per trajectory; Appendix C.2 reports positive rates for all targets.

Temporal distance probing. Given $\varphi(s_t)$ and $\varphi(s_{t'})$, predict the number of steps separating them, implemented as multi-class classification over discretized bins. This probes whether the representation preserves *temporal structure* globally rather than locally: two states that look similar in pixel space may be far apart in reachability, and a representation that collapses them will fail here while still supporting single-step predictions. Aytar et al. (2018); Ye et al. (2020) use this signal as a self-supervised pretraining objective. We explain the discretization of bins in detail in Appendix D.1.1.

Effective rank (Kumar et al., 2021). For the $N \times d$ feature matrix Φ evaluated over a batch of size N with embedding dimension d and $\delta \in (0, 1)$, report $\text{srank}_\delta(\Phi) = \min\left\{k : \frac{\sum_{i=1}^{k} \sigma_i(\Phi)}{\sum_{i=1}^{d} \sigma_i(\Phi)} \geq 1 - \delta\right\}$, where $\{\sigma_i(\Phi)\}$ are the singular values of Φ in decreasing order, i.e. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d$. A low effective rank indicates that the representation is concentrated in a small number of directions, consistent with *representational collapse*, whereas a higher rank indicates a richer set of active feature directions. We use $\delta = 0.01$, following the original work.

Further implementation details for each proxy can be found in Appendix D.1.

3.2 Normalized Fraction-of-Oracle

Rank correlation summarizes how well a proxy orders models across the full range of representation quality, but this is misaligned with the practical use case for proxies: model selection. Spearman's ρ fails to capture what matters here in two ways. First, ρ depends on ranks alone, so swapping the top two representations incurs the same penalty whether their downstream performances are nearly identical or vastly different. Second, ρ weighs rank discrepancies uniformly across all M representations, whereas selection depends only on the top. Away from $\rho = 1$, the value is uninformative about selection: a low ρ may reflect poor ordering among weak representations while the strong ones are correctly identified, and a high ρ is compatible with missing the best model.

The same tension arises in offline policy selection, where rank correlation is supplemented with metrics computed on the selection outcome, such as regret@k (Fu et al., 2021). We adopt this principle. Given proxy and downstream scores for M candidate representations, the top-k *normalized fraction-of-oracle* is the best downstream performance among the proxy's top-k picks, as a normalized fraction of the best achievable across all M:

$$\text{NFoO}_k \;=\; \frac{\max_{m \in \mathcal{M}_k} S(\varphi_m) - \min_{m \in [M]} S(\varphi_m)}{\max_{m \in [M]} S(\varphi_m) - \min_{m \in [M]} S(\varphi_m)},$$

where $\mathcal{M}_k$ indexes the k highest-scoring representations under the proxy and S is the downstream score. Higher values are better: 1 indicates that the proxy selected the best representation in the pool, 0 the worst, and 0.5 a representation midway between the two. When the minimum score in the pool is non-negative, this metric also provides a lower bound on the selected representation's performance as a fraction of the best achievable performance. We use human-normalized scores (HNS) for S, report $k = 1, 2, 3$ alongside ρ, and default to $k = 1$ unless stated otherwise.

4 Experimental Setup

Our experiments are designed to evaluate every proxy according both ρ and NFoO_k. Here we describe the experimental design: the data and pretraining methods used to produce candidate representations and the RL finetuning protocol used to evaluate them downstream.

Data. We experiment on 15 Atari games via the Arcade Learning Environment (ALE) (Bellemare et al., 2013; Machado et al., 2017), using data from RL Unplugged (Gulcehre et al., 2020), who collected transitions from five independent runs of an online Deep Q-Network (DQN) agent, each trained for 50 million transitions and divided into checkpoints of 1M environment steps.

We pretrain representations on a dataset of 1.5 million transitions from data collected early on in the DQN agent's training. The data used for probing is independent of the pretraining data; we report results primarily using a dataset of 100k transitions collected using the final DQN checkpoint,

Table 1: The 16 pretraining methods, named `Base[+Aux]-Transition-Size`. Each row lists the variants of one base method.

Base	Variants
ATC	-S, +I-S
ST-DIM	-S, +I-S
VAE-T	-S, +I-S
BYOL	+GI-CD-M, +G-GD-M, +GI-GD-M, +GI-GL-M
BT-0.7	-GL-M, +I-GL-M, +I-GL-L
BT-Rand	-GL-M, +I-GL-M, +I-GL-L

although we perform ablations using a variety of checkpoints and find that this choice is not load-bearing (see Appendix G.2). For further details regarding data choices and the exact list of games we use, refer to Appendix C.

Pretraining algorithms. We choose 16 pretraining methods to span contrastive, predictive, and generative paradigms in self-supervised learning: ATC (Stooke et al., 2021) and ST-DIM (Anand et al., 2019) (contrastive), several variants built upon BYOL (Grill et al., 2020) and Barlow Twins (Zbontar et al., 2021) (predictive), and VAE (Kingma & Welling, 2022) (generative). Following Zhang et al. (2024); Schwarzer et al. (2021b), these BYOL- and Barlow Twins-based variants augment the base self-supervised objective with a goal-conditioned RL loss and an inverse dynamics modeling loss as auxiliary objectives, and additionally train a transition model that is discarded after pretraining; the different variants correspond to different subsets of these auxiliary losses and different architectures used for the transition model. We experiment with three varying encoder sizes.

We name each pretrained model according to the scheme `Base[+Aux]-Transition-Size`. `Base` is the self-supervised objective (`ATC`, `ST-DIM`, `VAE-T`, `BYOL`, or `BT` for Barlow Twins), where `BT-0.7` and `BT-Rand` denote two variants of Barlow Twins. `Aux` lists the auxiliary objectives, `G` for the goal-conditioned RL loss and `I` for the inverse dynamics loss, omitted when no auxiliary loss is used. `Transition` is the transition model architecture (`CD`, `GD`, or `GL`), omitted for methods that train no transition model. `Size` is the encoder size (`S`, `M`, or `L`). The 16 models are listed in Table 1; additional details are given in Appendix B.

RL finetuning setup. Prior to online RL finetuning, we freeze the pretrained representations. This decouples representation learning from learning a policy (Stooke et al., 2021; Schwarzer et al., 2021b). We finetune a Rainbow DQN agent (Hessel et al., 2017) for 100k steps, consistent with the Atari100k benchmark (Łukasz Kaiser et al., 2020). Additionally, to assess how proxy validity generalizes to different learners, we finetune a PPO agent (Schulman et al., 2017) for 10 million steps. We normalize the finetuning scores using human normalized scores (HNS), defined as $\frac{\text{agent score} - \text{random score}}{\text{human score} - \text{random score}}$. All finetuning runs use 10 independent seeds. Learning rate choices are discussed in Appendix E.3.

Compute. We report GPUs used and median runtimes for pretraining, probing, and finetuning in Appendix F. Notably, computing all seven proxies on a representation is 400–$1{,}100\times$ cheaper than the 10-seed finetuning evaluation used to score it. This calculation assumes a single proxy seed suffices, which we validate in Appendix G.5 by showing proxy variance is very low.

Scope. To isolate proxy validity from other sources of variation, we restrict our study to Atari, pretrain and finetune each representation on the same game, and keep encoders frozen during finetuning. Our conclusions should be read within this scope; we discuss its consequences in Section 6.

5 Results

We organize our results as a sequence of selection problems. We begin with the setting closest to prior work: choosing among 16 pretraining methods, aggregated over 15 games and evaluated with Rainbow at 100k steps. We then vary its conditions one at a time, using a different downstream learner (PPO at 10M steps) and a single game rather than an average over many. We score each setting with both ρ and NFoO. Finally, we use proxies to detect when pretraining has plateaued, so that it can be stopped early.

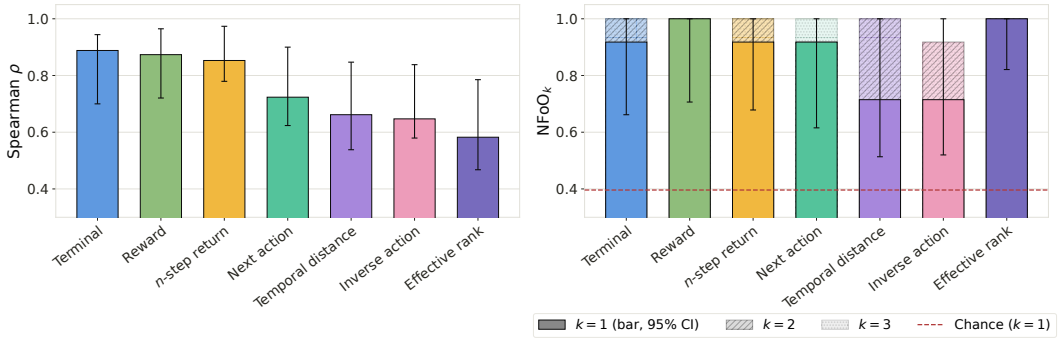

Figure 1: **Cross-method selection under Rainbow**. Spearman ρ (left) and normalized fraction-of-oracle (right) for the seven proxies over the 16 pretraining methods at their final checkpoints. Downstream score is the IQM of HNS over 150 seed-task runs (15 games $\times$ 10 seeds) after 100k steps of Rainbow finetuning; proxy score is the mean of per-game proxy scores over the same 15 games. Bars show $k = 1$ with 95% CIs bootstrapped over games; hatched extensions show $k \in \{2, 3\}$. The dashed line marks the expected NFoO_1 of selecting a method at random (0.39).

5.1 Cross-method selection

Establishing which of several pretraining methods is best requires finetuning each one across every game and seed. We first consider cross-method selection, following the setting of Zhang et al. (2024), to select the best pretraining method with proxies. We take a method's downstream score to be the interquartile mean (IQM) of HNS over the pool of all 150 seed-task runs, following Agarwal et al. (2021). Its proxy score is the mean of its per-game proxy scores across the same 15 games. Figure 1 reports both ρ and NFoO_k for $k \in \{1, 2, 3\}$.

Ranking quality varies across proxies; all of them select far above chance. Selecting a method at random recovers $\mathrm{NFoO}_1 = 0.39$, so the pool of methods is genuinely discriminative. Every proxy beats the random baseline by a wide margin. Rank correlation, however, spreads widely: terminal, reward, and n-step return order the pool well ($\rho \approx 0.85$ to 0.89), while inverse action, temporal distance, and effective rank trail at $\rho \approx 0.58$ to 0.66. Selection quality shows much less spread; five of seven proxies reach $\mathrm{NFoO}_1 \geq 0.91$, reward and effective rank select the oracle outright. By $k = 2$, every proxy reaches at least 0.91.

Use both metrics to guard against proxy-specific bias. Effective rank is an interesting example: it is the worst-ranked proxy by ρ and jointly the best by NFoO_1. Further investigation reveals that it ranks `ST-DIM-S` and `ST-DIM+I-S` 4th and 3rd among the 16 methods, yet their Rainbow downstream performance places them only 15th and 13th. These two methods account for most of the disagreement between effective rank and downstream performance: removing them raises the Spearman correlation from $\rho = 0.58$ to $\rho = 0.95$. ST-DIM's dense spatial contrastive objective plausibly yields high-rank features (which effective rank measures), but these features are not guaranteed to be task-relevant. Its perfect NFoO_1 is thus partly fortunate: had an ST-DIM variant ranked first, it would have selected a poor method (we analyze sensitivity of NFoO_k to pool composition in Appendix H). NFoO tells us how well a proxy selects from the pool at hand; ρ reveals the biases when the pool changes. We therefore recommend checking both before relying on a proxy.

The seven proxies rely on signals that differ, though to varying degrees, ranging from reward presence to episode structure to temporal distance, yet all select far above chance, and effective rank does so using no task labels at all. In this setting, good selection does not appear to depend on any particular target; a practitioner may use whichever proxy is computable under their data constraints.

5.2 Generalizing across downstream learners

Section 5.1 evaluates proxies against a single downstream learner, as does prior work. This leaves open whether a proxy validated against one learner remains valid for another: the representation is

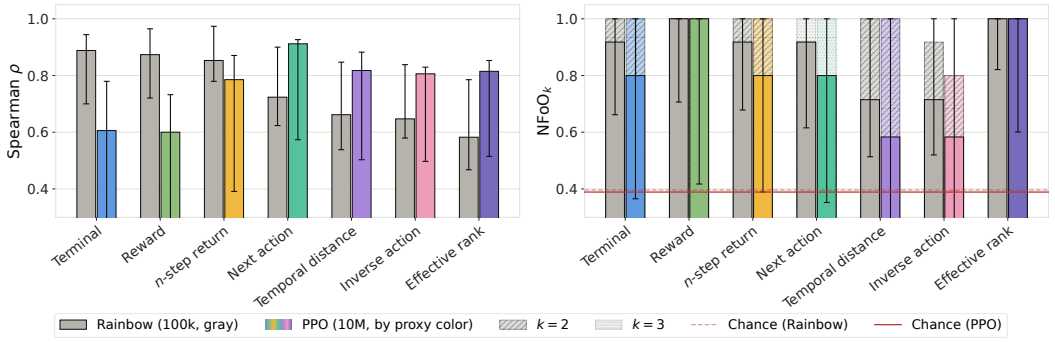

Figure 2: **Selection quality transfers across downstream learners; rank correlation does not.** The cross-method setting of Figure 1 repeated with a PPO agent finetuned for 10M steps (colored) alongside Rainbow at 100k (gray). Spearman ρ (left) and normalized fraction-of-oracle (right), with $k = 2$ and $k = 3$ hatched and a chance line for each learner. Proxies with high ρ under Rainbow do not always have high ρ under PPO (and vice versa), while NFoO_1 transfers well between learners.

fixed, but the learner built on top of it is not. We repeat the cross-method setting with a PPO agent finetuned for 10M steps, which changes both algorithm and budget, so we test generalization across learners as a whole rather than either factor in isolation. Figure 2 compares the two learners.

Selection quality transfers across learners; rank correlation does not. Under PPO, rank correlation reorders the proxies almost entirely: terminal and reward drop from among the strongest under Rainbow to the weakest, while the four weakest under Rainbow (next action, temporal distance, inverse action, and effective rank) become the strongest. Selection quality is far more stable: the ordering of proxies by NFoO_1 is largely preserved, with reward and effective rank selecting the oracle method under both learners. With a few more picks, the differences largely vanish: every proxy reaches $\mathrm{NFoO}_2 \geq 0.8$, and all but one reach 1.00 by $k = 3$. A practitioner using ρ under one learner to decide which proxy to trust would thus reach nearly the opposite conclusion from one using the other, whereas NFoO gives broadly the same answer under both. The learners disagree about the ordering of the pool ($\rho = 0.62$ between PPO and Rainbow IQM) but largely agree on which methods belong at the top, and selection depends only on the top.

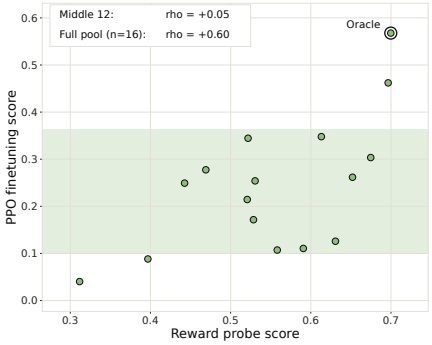

Figure 3: ρ **can understate selection quality.** Reward-probe score vs. PPO performance for the 16 pretraining methods.

We also see an example here of the difference between Spearman's ρ and NFoO (see figure 3). Under PPO, Spearman's ρ understates the selection quality of the reward probe: it correctly identifies the two best and two worst methods, ranking the best method under PPO first, but its ordering of the remaining 12 methods is uninformative ($\rho = 0.05$), dragging the overall correlation down to 0.60. In contrast, NFoO identifies reward as an effective proxy because it successfully selects the best method. Thus, both metrics should be considered when assessing whether a proxy is effective.

5.3 Per-game selection

The settings above score a method by its performance pooled over all 15 games, at its final pretraining checkpoint. We now vary both conditions. Each game is scored on its own, and every (method, checkpoint) pair counts as a distinct representation; due to compute constraints, we select 10 methods (Appendix A.2) and each of their 10 checkpoints, giving 100 total representations. Each candidate is finetuned with Rainbow for 100k environment steps and scored by HNS averaged over

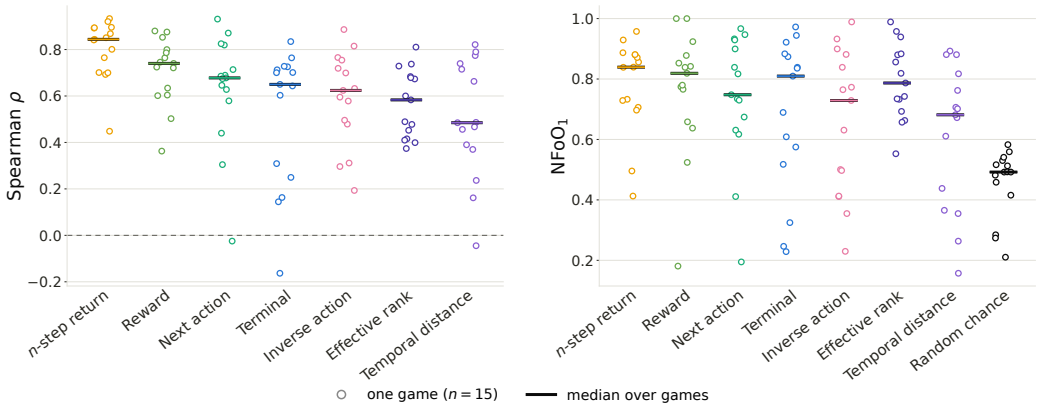

Figure 4: **Per-game selection over 100 candidate representations.** Spearman ρ (left) and NFoO_1 (right) computed within each game over a pool of 100 representations, finetuned with Rainbow for 100k steps and scored by HNS averaged over 10 seeds. Each circle is one of the 15 games; the horizontal bar is the median over games. The rightmost column on the right panel gives the per-game expected NFoO_1 of random selection. Per-game values are tabulated in Appendix A.2.

10 seeds. We report Spearman ρ and normalized fraction-of-oracle within each game, summarizing the 15 games by their median (Figure 4); per-game values are in Appendix A.2.

Proxies perform comparably and all beat chance at the representation level. Out of the 100 candidates, every proxy exceeds the random baseline in at least 12 of the 15 games (see tables in Appendix A.2). Selection quality is tightly clustered: the median NFoO spans only 0.68 to 0.84 across the seven proxies, against 0.48 to 0.84 for median ρ. Thus even at the representation level, proxy target does not dramatically affect model selection quality.

5.4 Early stopping of Pretraining

Section 5.3 shows that proxies select usefully within a single game. We now test their utility in early stopping: can proxies determine when to stop pretraining early, reducing compute while preserving the performance of the final checkpoint? In typical pretraining runs, later checkpoints perform similarly, so stopping once training enters this plateau would save compute. Our experiments show the same pattern: the earliest and final checkpoints are clearly separated, but many intermediate checkpoints are largely interchangeable (Appendix I.2).

Setup. We use a patience-based rule: once a proxy score has failed to improve on its best value by at least a fraction δ for p consecutive checkpoints, we halt and return the latest checkpoint. Performance is the chosen checkpoint's downstream score as a percentage of the final checkpoint's score (which can exceed 100%). We shift negative game scores to be positive so the ratio is well-defined, and report medians across (method, game) pairs. We compare against two baselines: (1) *fixed checkpoint*: halting at a preset checkpoint index, which simulates the common practice of training for a fixed budget and using the final checkpoint. Because intermediate checkpoints perform interchangeably, we expect this to be a strong baseline, though it requires choosing the budget in advance; (2) *validation loss*: the same patience-based rule applied to the pretraining loss on held-out data, treating a decrease as an improvement. Details are in Appendix A.3.

Proxy-guided stopping saves compute, and proxies behave interchangeably. Figure 5 shows results for patience values $p = 1, \ldots, 5$. Across all values, every proxy lies on or above the fixed-checkpoint curve; validation loss does as well, except at $p = 1$. At $p = 2$, for example, every proxy halts after only 40–50% of total compute and reaches 94-97.5% of final-checkpoint performance, compared to 92% for a fixed checkpoint at comparable compute. Validation loss reaches the full final-checkpoint score, but only by training substantially longer (to $\approx$70% of compute). Consistent with our findings elsewhere, the choice of proxy matters little. The proxy curves largely overlap, so practitioners can use whichever proxy is most convenient.

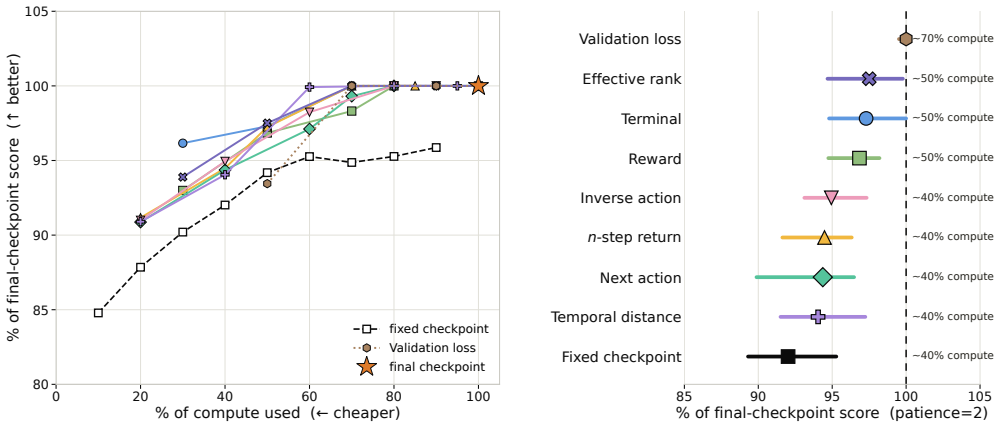

Figure 5: **Proxy-guided early stopping.** *Left:* Median percentage of the final checkpoint's score versus median percentage of pretraining compute used, for patience $p = 1, \ldots, 5$. The dashed line halts at fixed checkpoint indices. *Right:* Median % of final-checkpoint score reached by each proxy's patience-based stop ($p = 2$), with 95% bootstrapped confidence intervals. Annotations give the median compute used. In both panels, δ is fixed at 0.02.

5.5 Do proxies rank differently?

Sections 5.1–5.4 found that the choice of probing target has little effect on selection, which is meaningful only if the seven proxies produce different rankings. We test this using *shared error* (see Appendix D.3), the partial correlation between two proxy rankings after controlling for downstream score, which measures whether they misrank the same representations.

Two pairs of proxies are alike; the rest are not. Shared error is highest for next action and inverse action (0.80), and for reward and n-step return (0.63); no other pair exceeds 0.48, and the remaining pairs average 0.27. These pairs use closely related targets: reward presence is a binary transformation of the immediate reward, which also contributes to the n-step return, while next-action prediction can be viewed as inverse dynamics given only the first of its two observations. Outside of these nested pairs, the proxies are generally wrong about different representations, so their similar selection performance stands as an interesting result rather than a consequence of their being near-copies.

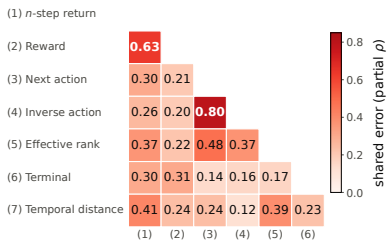

Figure 6: **Shared error between proxies.** Partial Spearman correlation between each pair of proxy rankings with the downstream score removed, computed within each game over the 16 methods at their final checkpoints and averaged over the 15 games. Replication over the larger pool of 100 representations is given in Appendix D.3.

6 Discussion

In this work, we present an asymmetry between rank correlation and model selection via experimenting with seven proxies, and we observe it across every selection scenario we study. With NFoO, we discovered the choice of proxy is stable for model-selection purpose. For a complete evaluation of proxies, we recommend reporting both ρ and NFoO. Future work remains to establish whether our results extend beyond Atari, to encoders that are updated during RL, and to transfer across tasks. Finally, because the proxies select similarly and their ability to order candidates shifts with the downstream learner, we cannot say whether a universally best proxy exists. Finding one, or showing that none exists, is an open question for future work.

AI USE STATEMENT

In this work, we used generative AI tools for implementing a large portion of our codebase and submitting experiments. We have not used generative AI tools for designing the experiments or interpreting results. Additionally, we used generative AI tools to polish this manuscript and write the code used to generate figures and tables. We have reviewed all AI-generated code or text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We release code at `https://anonymous.4open.science/r/rl-representation-proxies-DE70/` for pretraining all representations, training and evaluating all probes, finetuning with both Rainbow and PPO, and reproducing every data-derived table and figure in the paper.

REFERENCES

Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. Zero-cost proxies for lightweight nas, 2021. URL https://arxiv.org/abs/2101.08134.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. 2020.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018. URL https://arxiv.org/abs/1610.01644.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. *Advances in neural information processing systems*, 32, 2019.

Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando De Freitas. Playing hard exploration games by watching youtube. *Advances in neural information processing systems*, 31, 2018.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning, 2017. URL https://arxiv.org/abs/1707.06887.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL https://arxiv.org/abs/1412.3555.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2126–2136, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Andrea Dittadi, Frederik Träuble, Manuel Wüthrich, Felix Widmaier, Peter Gehler, Ole Winther, Francesco Locatello, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. The role of pre-trained representations for the ood generalization of reinforcement learning agents, 2022. URL https://arxiv.org/abs/2107.05686.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad 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 Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, Ziyu Wang, Alexander Novikov, Mengjiao Yang, Michael R. Zhang, Yutian Chen, Aviral Kumar, Cosmin Paduraru, Sergey Levine, and Tom Le Paine. Benchmarks for deep off-policy evaluation, 2021. URL https://arxiv.org/abs/2103.16596.

Yuwei Fu, Di Wu, and Benoit Boulet. A closer look at offline rl agents. *Advances in Neural Information Processing Systems*, 35:8591–8604, 2022.

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, 2020. URL https://arxiv.org/abs/2006.07733.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in neural information processing systems*, 33:7248–7259, 2020.

Caglar Gulcehre, Srivatsan Srinivasan, Jakub Sygnowski, Georg Ostrovski, Mehrdad Farajtabar, Matt Hoffman, Razvan Pascanu, and Arnaud Doucet. An empirical study of implicit regularization in deep offline rl, 2022. URL https://arxiv.org/abs/2207.02099.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022. URL https://arxiv.org/abs/2010.02193.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters, 2019. URL https://arxiv.org/abs/1709.06560.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning, 2017. URL https://arxiv.org/abs/1710.02298.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks, 2016. URL https://arxiv.org/abs/1611.05397.

Donghu Kim, Hojoon Lee, Kyungmin Lee, Dongyoon Hwang, and Jaegul Choo. Investigating pre-training objectives for generalization in vision-based reinforcement learning, 2024. URL https://arxiv.org/abs/2406.06037.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning, 2021. URL https://arxiv.org/abs/2010.14498.

Hojoon Lee, Koanho Lee, Dongyoon Hwang, Hyunho Lee, Byungkun Lee, and Jaegul Choo. On the importance of feature decorrelation for unsupervised representation learning in reinforcement learning. In *International Conference on Machine Learning*, pp. 18988–19009. PMLR, 2023.

Felix Leibfried and Peter Vrancx. Model-based regularization for deep reinforcement learning with transcoder networks, 2018. URL https://arxiv.org/abs/1809.01906.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning, 2022. URL https://arxiv.org/abs/2204.09560.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks, 2023. URL https://arxiv.org/abs/2303.01486.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents, 2017. URL https://arxiv.org/abs/1709.06009.

Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training, 2021. URL https://arxiv.org/abs/2006.04647.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb 2015. doi: 10.1038/nature14236.

Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning, 2020. URL https://arxiv.org/abs/2007.09055.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017. URL https://arxiv.org/abs/1705.05363.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. URL https://arxiv.org/abs/1801.04381.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations, 2021a. URL https://arxiv.org/abs/2007.05929.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021b.

Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning, 2017. URL https://arxiv.org/abs/1612.07307.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 32145–32168. PMLR, 2023.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020. URL https://arxiv.org/abs/2004.04136.

Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch, 2019. URL https://arxiv.org/abs/1909.01500.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International conference on machine learning*, pp. 9870–9879. PMLR, 2021.

Adrien Ali Taiga, Rishabh Agarwal, Jesse Farebrother, Aaron Courville, and Marc G Bellemare. Investigating multi-task pretraining and generalization in reinforcement learning. In *The eleventh international conference on learning representations*, 2023.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations, 2019. URL https://arxiv.org/abs/1905.06316.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2016. URL https://arxiv.org/abs/1511.06581.

Joel Ye, Dhruv Batra, Erik Wijmans, and Abhishek Das. Auxiliary tasks speed up learning pointgoal navigation, 2020. URL https://arxiv.org/abs/2007.04561.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. URL https://arxiv.org/abs/2103.03230.

Wancong Zhang, Anthony GX-Chen, Vlad Sobal, Yann LeCun, and Nicolas Carion. Light-weight probing of unsupervised representations for reinforcement learning. *Reinforcement Learning Journal*, 4:1924–1949, 2024.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.

A NUMERICAL RESULTS

The following tables report the numerical values underlying the figures in Section 5.

A.1 CROSS-METHOD SELECTION

Here we give tabulated results regarding Sections 5.1 and 5.2. The point estimates of Figures 1 and 2 are given in Table 2 (Spearman ρ) and Table 3 (NFoO_k).

Table 2: Cross-method Spearman correlation between each proxy and downstream finetuning performance. Computed over the pool of the final checkpoints of all 16 pretraining methods; downstream score is HNS, IQM'd over the pooled 10-seed $\times$ 15-game set.

Task	Rainbow ρ	p	PPO ρ	p
Terminal	$+0.888$	< 0.001	$+0.606$	0.013
Reward	$+0.874$	< 0.001	$+0.600$	0.014
n-step return	$+0.853$	< 0.001	$+0.785$	< 0.001
Next action	$+0.724$	0.002	$+0.912$	< 0.001
Temporal distance	$+0.662$	0.005	$+0.818$	< 0.001
Inverse action	$+0.647$	0.007	$+0.806$	< 0.001
Effective rank	$+0.582$	0.018	$+0.815$	< 0.001

Table 3: Cross-method NFoO_k. Computed over the pool of the final checkpoints of all 16 pretraining methods; downstream score is HNS, IQM'd over the pooled 10-seed $\times$ 15-game set. Random = expected NFoO_k of k uniformly drawn methods.

Task	Top pick	Rainbow NFoO_k			PPO NFoO_k		
		k=1	k=2	k=3	k=1	k=2	k=3
Reward	`BT-Rand+I-GL-L`	1.000	1.000	1.000	1.000	1.000	1.000
Effective rank	`BT-Rand+I-GL-L`	1.000	1.000	1.000	1.000	1.000	1.000
Terminal	`BT-0.7+I-GL-L`	0.918	1.000	1.000	0.800	1.000	1.000
n-step return	`BT-0.7+I-GL-L`	0.918	1.000	1.000	0.800	1.000	1.000
Next action	`BT-0.7+I-GL-L`	0.918	0.918	1.000	0.800	0.800	1.000
Inverse action	`BYOL+GI-GD-M`	0.715	0.918	0.918	0.583	0.800	0.800
Temporal distance	`BYOL+GI-GD-M`	0.715	1.000	1.000	0.583	1.000	1.000
Random	—	0.396	0.576	0.689	0.389	0.543	0.630

A.2 PER-GAME SELECTION

In this section, we present the results on the use of proxies to select the best representation for each game. We treat each (pretraining method, pretraining checkpoint) pair as a distinct representation, rather than scoring methods at their final checkpoint alone. This gives 100 candidates per game: 10 methods at 10 checkpoints each (`BT-Rand+I-GL-M`, `BT-0.7+I-GL-M`, `BT-0.7-GL-M`, `BYOL+GI-GD-M`, `BYOL+GI-CD-M`, `BT-0.7+I-GL-L`, `BT-Rand+I-GL-L`, `ATC-S`, `ST-DIM-S`, `VAE-T-S`). The same 100 representations are available for all 15 games.

Each candidate is finetuned with Rainbow for 100k environment steps and scored by human-normalized score averaged over 10 seeds. PPO is absent since its runs exist only at each method's final checkpoint, so it has no checkpoint axis to score. The learning rate is chosen once per (method, game), by taking the rate with the highest 10-seed mean at that method's *own final* checkpoint, and is then reused for all ten of its checkpoints (see Table 26 for the exact learning rates used).

We compute both Spearman ρ values and NFoO_k for $k = 1, 2, 3$ within each game and also summarize the 15 games via median; refer to Tables 4 and 5.

Table 4: Per-game Spearman ρ between proxy score and Rainbow finetuning score, over the 100-representation pool. Each column is one proxy; each row is one game. The last row is the median over the 15 games.

Game	Terminal	Reward	n-step return	Next action	Inverse action	Temporal distance	Effective rank
ms_pacman	0.73	0.74	0.84	0.83	0.81	0.72	0.68
assault	0.73	0.76	0.84	0.93	0.77	0.48	0.81
alien	0.70	0.60	0.85	0.87	0.89	0.77	0.49
breakout	0.83	0.88	0.90	0.69	0.31	0.79	0.74
seaquest	0.70	0.88	0.92	0.67	0.70	0.46	0.68
boxing	0.76	0.85	0.93	0.69	0.75	0.24	0.60
qbert	0.65	0.63	0.89	0.44	0.63	0.82	0.73
demon_attack	0.60	0.72	0.69	0.82	0.60	0.74	0.37
asterix	0.71	0.75	0.89	0.58	0.48	0.66	0.40
gopher	0.31	0.80	0.76	0.71	0.62	0.47	0.48
pong	0.25	0.72	0.87	0.63	0.50	0.37	0.67
krull	0.16	0.79	0.80	0.65	0.72	0.16	0.41
kung_fu_master	0.14	0.36	0.45	0.68	0.58	0.39	0.58
amidar	-0.16	0.50	0.70	0.30	0.30	0.49	0.45
frostbite	0.64	0.60	0.70	-0.02	0.19	-0.04	0.42
Median	**0.65**	**0.74**	**0.84**	**0.68**	**0.62**	**0.48**	**0.58**

Table 5: Per-game NFoO_k over the 100-representation pool. Each cell gives **top-1/top-2/top-3**. *Random* is the expected top-1 NFoO_1 from drawing one representation uniformly at random. The last row is the median over the 15 games.

Game	Terminal	Reward	n-step return	Next action	Inverse action	Temporal distance	Effective rank	Random
ms_pacman	.32/.94/.94	.92/.92/.94	.73/.91/.91	.84/.84/.89	.73/.91/.91	.61/.61/.61	.94/.95/.95	.49
assault	.81/.81/.81	.84/.90/.90	.84/.86/.86	.75/.93/.93	.93/.93/.93	.44/.48/.48	.79/.82/.82	.42
alien	.84/.84/.84	.85/.97/.97	.84/.95/.95	.97/.97/.97	.84/.97/.97	.76/.96/1	.55/.55/.65	.49
breakout	.97/1/1	1/1/1	.71/.73/.73	.93/.97/.97	.35/.35/.69	.35/.35/.35	.69/.69/.73	.21
seaquest	.69/.69/.69	.82/1/1	.88/.88/1	.63/.69/.69	.63/.69/.69	.88/.88/1	.82/.88/1	.46
boxing	.94/.96/1	.77/.77/.77	.96/1/1	.82/.86/.86	.50/.60/.60	.82/.90/.90	.96/1/1	.51
qbert	.92/1/1	.66/.81/.81	.93/.93/.93	.93/.93/.99	.99/.99/.99	.70/.78/.86	.99/.99/.99	.56
demon_attack	.61/.78/.85	.18/.18/.42	.41/.42/.47	.73/.73/.80	.41/.47/.47	.37/.37/.44	.73/.80/.84	.27
asterix	.25/.44/.44	.77/.78/.89	.89/.89/.89	.19/.29/.29	.23/.23/.23	.88/.88/.88	.66/.70/.71	.28
gopher	.52/.54/.61	1/1/1	.50/.50/.54	.95/.95/.96	.50/.67/.79	.67/.67/.67	.66/.75/1	.48
pong	.57/.65/.69	.64/.85/.91	.87/.87/.87	.73/1/1	.76/.90/.90	.68/.68/.68	.74/.80/.80	.52
krull	.87/.87/.88	.88/.88/.93	.86/.87/.90	.90/.90/.95	.90/.95/.95	.71/.77/.88	.86/.88/.88	.58
kung_fu_master	.88/.89/.89	.78/.78/.88	.70/.95/.95	.62/.75/.88	.88/.88/.88	.89/.89/.89	.88/.88/.88	.54
amidar	.23/.23/.56	.84/.84/.84	.73/.81/.85	.41/.41/.46	.41/.41/.43	.26/.36/.36	.73/.74/.74	.53
frostbite	.84/.99/.99	.52/.68/.99	.88/.88/.88	.67/.67/.67	.77/.91/.91	.16/.16/.16	.88/.88/1	.49
Median	**.81/.84/.85**	**.82/.85/.91**	**.84/.88/.89**	**.75/.86/.89**	**.73/.88/.88**	**.68/.68/.68**	**.79/.82/.88**	**.49**

A.3 Early Stopping

Table 6 gives the values underlying the left panel of Figure 5. Larger p requires more evidence of stagnation before halting, so both compute and percentage of final-checkpoint performance generally increase for every proxy.

Early stopping rule. We apply a patience-based stopping rule independently to each (method, game) pretraining run. Let $q_1, \ldots, q_{10}$ denote the signal values at the run's ten checkpoints, oriented so that higher is better. For validation loss, we negate the loss before applying the rule. We maintain a reference value b_i and a patience counter c_i, initialized as $b_1 = q_1$ and $c_1 = 0$.

For $i = 2, \ldots, 10$, checkpoint i satisfies the threshold condition if

$$q_i - b_{i-1} \geq \delta r_i,$$

Table 6: Early-stopping patience sweep. Each cell is median compute% / median % of final-checkpoint score over 150 qualifying (method, game) cells (10 early-stopping-eligible methods $\times$ 15 games). The fixed-checkpoint row reports the closest-matching index, which varies by column (p=1: idx 2, p=2: idx 4, p=3: idx 6, p=4: idx 8, p=5: idx 10). Random checkpoint selection reaches 93% of final-checkpoint score at 55% compute. A proxy that never triggers a stop within the checkpoint budget falls back to the final checkpoint (index 10).

Proxy	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
Terminal	30 / 96%	50 / 97%	70 / 100%	100 / 100%	100 / 100%
Effective rank	30 / 94%	50 / 98%	70 / 100%	90 / 100%	100 / 100%
Reward	30 / 93%	50 / 97%	70 / 98%	80 / 100%	100 / 100%
n-step return	20 / 91%	40 / 94%	50 / 97%	70 / 100%	85 / 100%
Inverse action	20 / 91%	40 / 95%	60 / 98%	80 / 100%	90 / 100%
Temporal distance	20 / 91%	40 / 94%	60 / 100%	80 / 100%	95 / 100%
Next action	20 / 91%	40 / 94%	60 / 97%	70 / 99%	80 / 100%
Validation loss	50 / 93%	70 / 100%	90 / 100%	100 / 100%	100 / 100%
Fixed checkpoint (compute-matched)	20 / 88%	40 / 92%	60 / 95%	80 / 95%	100 / 100%

where δ is a threshold parameter and $r_i \geq 0$ is a signal-specific reference distance defined below. We use $\delta = 0.02$ by default and examine its sensitivity in Appendix I.1.

When the condition is satisfied, we update the reference value to $b_i = q_i$ and reset the counter to $c_i = 0$. Otherwise, we retain $b_i = b_{i-1}$ and increment the counter, $c_i = c_{i-1}+1$. Thus, b_i is updated only when the threshold condition is satisfied; it is not necessarily the maximum of all previously observed scores.

We halt at the first checkpoint $i^\star$ for which $c_{i^\star} = p$ and return that checkpoint. If the counter never reaches p, we return the final checkpoint. When $r_i = 0$, the condition reduces to $q_i \geq b_{i-1}$, so an unchanged score also resets the counter.

Reference distance. The seven proxies and validation loss differ in scale and interpretation. We use a common threshold parameter δ, with the following signal-specific reference distances:

- For terminal, n-step return, next action, inverse action, and temporal distance,
$$r_i = \max(q_i, 0),$$
using zero as the reference value.
- For reward,
$$r_i = \max(q_i - 0.5, 0),$$
using a fixed reference value of 0.5.
- For effective rank,
$$r_i = \max(d - q_i, 0),$$
using the remaining distance to the representation dimension d.
- For validation loss, with q_i denoting the negated loss,
$$r_i = \max_{j \leq i} q_j - \min_{j \leq i} q_j,$$
using the range of signal values observed up to the current checkpoint.

We report $p \in \{1, \ldots, 5\}$ at the default δ and sweep $\delta \in \{0.5\%, 1\%, 2\%, 4\%, 8\%\}$ at $p = 2$ in Appendix I.1.

Percentage of final checkpoint score. We report the selected checkpoint's raw finetuning score as a percentage of the final checkpoint's raw score, rather than the min-max normalized NFoO_k used elsewhere in the paper, so that the y-axis has a direct reading: "how much of full training you kept." Because raw scores can be negative (e.g. Pong), each game's scores are first shifted by a constant so that its pooled minimum across all methods, checkpoints, and seeds is exactly zero, leaving the ratio well-defined and the final checkpoint at exactly 100% by construction.

Validation loss baseline. The validation loss baseline is each method's own pretraining objective evaluated on a held-out set of 100k transitions drawn from run 2 of RL Unplugged Atari at checkpoints 3–5, matching the checkpoint indices of the pretraining data but from a separate run. Because each method's loss is on its own scale, the baseline is only ever compared within a single pretraining run.

B Pretraining Methods

In this section we provide more details regarding each of the pretraining methods we consider. Section B.1 lists every pretraining method we use and states in which experiments it is used and Section B.2 describes each of the pretraining components and their hyperparameters.

B.1 List of pretraining methods

The naming convention is `Base[+Aux]-Transition-Size`, where the transition model is omitted when not used; see Section 4 and Table 1 for details. Here we list every method explicitly. The final pretrained checkpoints of all 16 methods are used in the cross-method selection experiments in Sections 5.1 and 5.2. When finetuning every checkpoint, however, we restrict to a set of 10 methods due to computational constraints; we denote the methods with all checkpoints finetuned (used in Sections 5.3 and 5.4) with $(\star)$.

- `ATC-S` $(\star)$
- `ATC+I-S`
- `ST-DIM-S` $(\star)$
- `ST-DIM+I-S`
- `VAE-T-S` $(\star)$
- `VAE-T+I-S`
- `BYOL+GI-CD-M` $(\star)$
- `BYOL+GI-GD-M` $(\star)$
- `BYOL+G-GD-M`
- `BYOL+GI-GL-M`
- `BT-0.7-GL-M` $(\star)$
- `BT-0.7+I-GL-M` $(\star)$
- `BT-0.7+I-GL-L` $(\star)$
- `BT-Rand-GL-M`
- `BT-Rand+I-GL-M` $(\star)$
- `BT-Rand+I-GL-L` $(\star)$

B.2 Pretraining components

We describe in additional detail the base objectives, auxiliary losses, transition models, and encoder models. In Table 7 we list pretraining hyperparameters shared among all methods.

Throughout this section we abbreviate the representation $\varphi(s_t)$ as e_t, writing $\tilde{e}_t$ for target-encoder representations and $\hat{e}_t$ for those produced by the transition model.

Table 7: Pretraining settings shared across methods except where noted.

	Setting	Value
Optim.	Optimizer	Adam
	Learning Rate	10^{-4} (except ATC: 10^{-3})
	Adam ϵ	1.5×10^{-4}
	Weight decay	1×10^{-6}
	Gradient-norm clip	10
	Epochs	20
	Checkpoint interval	every 2 epochs
	Batch size	64 (except ATC: 512)
Other	Projection	Q-head first layers (reused)
	Projection output dim	1024
	w_{rl}, w_{bc}, w_{t_0}	0, 0, 0
	Seeds per (method, game)	1

B.2.1 BASE OBJECTIVES

The base objectives we rely on are BYOL (Grill et al., 2020), Barlow Twins (Zbontar et al., 2021), ATC (Stooke et al., 2021), ST-DIM (Anand et al., 2019), and VAE-T (Kingma & Welling, 2022). The implementations for BYOL and Barlow Twins were taken from Zhang et al. (2024), and ATC, ST-DIM, and VAE-T from Stooke & Abbeel (2019).

Following standard practice in self-supervised learning, the objectives are computed on a lower-dimensional projection $y = \mathrm{proj}(e)$ rather than on the representation e itself, so that e can retain features the objective does not directly reward. Following Zhang et al. (2024), the projection reuses the first linear layer of each of the Q-head's advantage and value streams, concatenated to give 1024 dimensions.

We provide hyperparameters specific to each base objective in Table 8.

Table 8: Hyperparameters specific to each base objective.

Objective	Parameter	Value
BYOL	Predictor q	linear
	EMA coefficient τ	0.01
	Observation augmentation	shift, intensity
Barlow Twins	Off-diagonal weight λ	0.0051
	Balance coefficient λ_{bal}	0.7
	Observation augmentation	shift, intensity
ATC	Temporal offset k	1
	Predictor	residual MLP, width 512 + bilinear W
	Negatives per anchor	511 (batch size 512)
	Observation augmentation	shift
ST-DIM	Temporal offset	1
	Local feature map size	$64 \times 9 \times 9$
	Observation augmentation	—
VAE-T	Latent dimension d_z	128
	Action embedding dimension	128
	Decoder	transposed-conv mirror of encoder
	Observation augmentation	—

BYOL. Following Schwarzer et al. (2021a), we adapt BYOL (Grill et al., 2020) to a temporal, rather than augmentation-based, self-predictive objective. Given a representation e_t at time t, an online transition model rolls forward K steps conditioned on the actions taken, producing predicted representations $\hat{e}_{t+1}, \ldots, \hat{e}_{t+K}$. In parallel, a target encoder directly encodes the true observations visited at each of those future time steps, producing target representations $\tilde{e}_{t+1}, \ldots, \tilde{e}_{t+K}$. The target encoder is an exponential moving average of the online encoder, with gradients blocked. Both the predicted and target representations are projected to a lower-dimensional space, yielding $\hat{y}_{t+1:t+K}$ and $\tilde{y}_{t+1:t+K}$, and a linear predictor q maps from the online projection space to the target projection space. The loss is the negative cosine similarity between $q(\hat{y}_{t+k})$ and $\tilde{y}_{t+k}$, averaged over the K-step rollout:

$$\mathcal{L}_\theta^{\text{BYOL}}(\hat{y}_{t+1:t+K}, \tilde{y}_{t+1:t+K}) = -\frac{1}{K}\sum_{k=1}^{K} \frac{q(\hat{y}_{t+k}) \cdot \tilde{y}_{t+k}}{\|q(\hat{y}_{t+k})\|_2 \, \|\tilde{y}_{t+k}\|_2}.$$

Barlow Twins. Related to BYOL, we also consider Barlow Twins (Zbontar et al., 2021) as a prediction objective. Barlow Twins also employs an online and a target network; however, it regularizes the cross-correlation matrix between the two networks' projected predictions towards the identity, rather than using the predictor network q of BYOL. Intuitively, this encourages each dimension of the projection to be predictive of its counterpart while discouraging redundancy across dimensions. We follow the adaptation of Zhang et al. (2024) to the temporal setting by computing the cross-correlation matrix jointly across the batch and rollout-length dimensions, so that the resulting loss

combines an invariance term with a de-correlation term, weighted by a hyperparameter λ:

$$\ell(\hat{y}, \tilde{y}) = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2,$$

where C is the cross-correlation matrix between the projected outputs of the online and target branches,

$$C_{ij} = \frac{\sum_{b,t} \hat{y}_{b,t,i}\, \tilde{y}_{b,t,j}}{\sqrt{\sum_{b,t} \hat{y}_{b,t,i}^2} \sqrt{\sum_{b,t} \tilde{y}_{b,t,j}^2}},$$

with b indexing batch samples, t indexing time, and i, j indexing dimensions of the projection output. We set $\lambda = 0.0051$, following Zhang et al. (2024).

We apply the Barlow Balancing technique of Zhang et al. (2024), which forms a convex combination of two copies of the loss, each with one branch detached:

$$\mathcal{L}_\theta^{\text{BT}} = \lambda_{\text{bal}}\, \ell(\hat{y}, \text{sg}[\tilde{y}]) + (1 - \lambda_{\text{bal}})\, \ell(\text{sg}[\hat{y}], \tilde{y})\,,$$

where $\text{sg}[\cdot]$ is the stop-gradient operator. This reweights how much gradient each branch receives, which matters when the two branches share a trunk and the transition model trains more slowly than the encoder. Both variants use $\lambda_{\text{bal}} = 0.7$ (hence the suffix 0.7), and differ only in the construction of the target branch: `BT-0.7` shares the encoder between the online and target branches, while `BT-Rand` uses a separately initialized, frozen copy. Since `BT-Rand`'s target features are already detached, only the first term contributes gradient for that variant, though it remains scaled by λ_{bal}.

ATC. Augmented Temporal Contrast (Stooke et al., 2021) is a contrastive objective that combines data augmentation with temporal prediction. Given an anchor observation s_t, ATC trains an encoder such that a randomly augmented view of s_t can be discriminated from a batch of negative examples when compared against a temporally offset, augmented view of s_{t+k}. Concretely, the encoded anchor is passed through a residual predictor network to produce $\hat{e}_{t+k}$, which is scored against the true encoded target $\tilde{e}_{t+k}$ (produced by an EMA target encoder) under a bilinear similarity $\text{sim}(\hat{e}, \tilde{e}) = \hat{e}^\top W \tilde{e}$ with learned W. The InfoNCE loss discriminates the true target from a set of negatives $\mathcal{N}$ drawn from other time steps and trajectories in the batch:

$$\mathcal{L}_\theta^{\text{ATC}} = -\log \frac{\exp \text{sim}(\hat{e}_{t+k}, \tilde{e}_{t+k})}{\exp \text{sim}(\hat{e}_{t+k}, \tilde{e}_{t+k}) + \sum_{e^- \in \mathcal{N}} \exp \text{sim}(\hat{e}_{t+k}, e^-)}.$$

ST-DIM. Spatiotemporal Deep InfoMax (Anand et al., 2019) is a contrastive objective that maximizes mutual information between representations of consecutive observations at both a global and a local (spatial) level. Given consecutive observations s_t and s_{t+1}, ST-DIM combines two InfoNCE-style terms. The global-local term scores the global representation of o_t against the spatially resolved local feature map of s_{t+1}, and the local-local term scores the local features of s_t against those of s_{t+1} at matching spatial positions. In both cases negatives are drawn from other time steps in the batch. Writing g_t for the global representation, $l_t^{(u,v)}$ for the local feature at spatial position (u, v), and $\text{sim}(\cdot, \cdot)$ for a bilinear similarity, the objective is

$$\mathcal{L}_\theta^{\text{ST-DIM}} = -\sum_{u,v} \left[\log \frac{\exp \text{sim}(g_t, l_{t+1}^{(u,v)})}{\sum_{t' \in \mathcal{N}} \exp \text{sim}(g_t, l_{t'}^{(u,v)})} + \log \frac{\exp \text{sim}(l_t^{(u,v)}, l_{t+1}^{(u,v)})}{\sum_{t' \in \mathcal{N}} \exp \text{sim}(l_t^{(u,v)}, l_{t'}^{(u,v)})} \right].$$

This encourages the representation to capture information that is consistent across nearby time steps at both the frame and patch level, without requiring an explicit forward or transition model.

VAE-T. As a generative alternative to the contrastive and self-predictive objectives above, we adapt a variational autoencoder (Kingma & Welling, 2022) to our temporal setting. The encoder output h_t is summed element-wise with a learned embedding a_t of the action taken at time t, and a linear head maps $h_t + a_t$ to the mean and log-variance of a Gaussian posterior over a latent z. A reparameterized sample from this posterior is decoded, via a transposed convolution mirror of the encoder, into a prediction of the next observation s_{t+1}. The training objective combines a reconstruction term, treating pixel intensities as Bernoulli targets and computed as a binary cross-entropy, with a KL-

divergence term regularizing the posterior toward a standard normal prior:

$$\mathcal{L}_\theta^{\text{VAE-T}} = \underbrace{\frac{1}{BC}\sum_{b,c}\sum_{u,v}\mathrm{BCE}(\hat{s}_{t+1}[b,c,u,v],\ s_{t+1}[b,c,u,v])}_{\text{reconstruction}} + \underbrace{\frac{1}{2B}\sum_{b}\sum_{j=1}^{d_z}\left(\mu_{bj}^2+\sigma_{bj}^2-\log\sigma_{bj}^2-1\right)}_{\text{KL}},$$

where B is the batch size, C the number of frame-stack channels, u, v index spatial positions, and μ_b, σ_b are the predicted posterior mean and standard deviation for sample b.

B.2.2 AUXILIARY LOSSES

The two auxiliary losses we consider are inverse action prediction (inverse dynamics) and goal-conditioning, both following Zhang et al. (2024); Schwarzer et al. (2021b). When a method uses auxiliary losses, the total pretraining objective is

$$\mathcal{L} = \mathcal{L}^{\text{base}} + \lambda_I \mathcal{L}^{\text{inv}} + \lambda_G \mathcal{L}^{\text{goal}},$$

with $\lambda_I = 1$ and $\lambda_G = 1$; methods using one or zero auxiliary loss set the necessary coefficient to zero. Hyperparameters for these auxiliary losses are given in Table 9.

Table 9: Hyperparameters for the auxiliary losses.

Loss	Parameter	Value
Inverse action	Weight λ_I	1
	Inverse model architecture	2-layer MLP, hidden width 256
Goal-conditioning	Weight λ_G	1
	Goal window W	50
	Permutation probability p_{perm}	0.2
	Noise weight η	0.5
	Distance sharpness	$\lambda = 5$
	Reward scale κ	10
	n-step	1
	Discount γ	0.99
	Support size N, range	$N = 51$, $[-10, 10]$

Inverse action loss. Given the representations of two consecutive observations, we predict the action taken between them. In both cases below, the inverse model consumes the projected embeddings y rather than the raw representation; we write $y = \mathrm{proj}(e)$. The two projections are concatenated and passed through a two-layer MLP with hidden width 256 whose softmax output parameterizes a distribution over the $|\mathcal{A}|$ discrete actions, and the loss is the cross-entropy against the true action. The representation supplying the later timestep is always detached, so the gradient trains only the branch encoding the earlier timestep.

The two forms differ in where the later representation comes from.

Methods with a transition model (`BYOL`, `BT`) compare the K-step rollout against target-encoder latents. The online branch supplies the predicted projection $\hat{y}_{t+k}$, the target branch supplies $\tilde{y}_{t+k+1}$, and the loss averages over the rollout:

$$\mathcal{L}_\theta^{\text{inv}} = -\frac{1}{K}\sum_{k=0}^{K-1}\log p_\theta(a_{t+k} \mid \hat{y}_{t+k}, \mathrm{sg}[\tilde{y}_{t+k+1}]),$$

where $\mathrm{sg}[\cdot]$ is the stop-gradient operator.

Methods without a transition model (`ATC`, `ST-DIM`, `VAE-T`) have no imagined rollout to draw from, so the loss reduces to a single real transition. Both observations o_t and o_{t+1} are independently

augmented and encoded by the *same online encoder*; the second is then detached. Writing y_t and y_{t+1} for the resulting projections,

$$\mathcal{L}_\theta^{\text{inv}} = -\log p_\theta(a_t \mid y_t, \text{sg}[y_{t+1}]) .$$

This is the $K = 1$ case of the expression above, with the target encoder replaced by a detached copy of the online encoder.

Goal-conditioning loss. Following Schwarzer et al. (2021b), we use a hindsight-style goal-reaching objective: a future observation is treated as a goal, the decrease in latent distance to that goal supplies a pseudo-reward, and a separate goal-conditioned head is trained with the same distributional Q-learning loss used for downstream RL. Unlike the inverse loss above, this objective operates on the raw convolutional feature maps rather than the projection-head embeddings.

Goals are sampled from the same trajectory by drawing an offset within a window of W steps and encoding the resulting observation with the frozen target encoder, $g_0 = \tilde{\varphi}(s_{t+\Delta})$, $\Delta \sim \text{Unif}\{1, \dots, W\}$. To diversify the goal distribution, sampled goals are permuted across the batch with probability p_{perm} and then interpolated toward a random unit-norm Gaussian direction u with a per-sample weight $w \sim \text{Unif}(0, \eta)$,

$$g = \frac{w\, u + (1 - w)\, g_0}{\|w\, u + (1 - w)\, g_0\|_2}.$$

The latent distance between a representation and a goal is

$$d(e, g) = 1 - \exp\!\big(-\lambda \|\bar{e} - \bar{g}\|_2^2\big) , \qquad \lambda = 5,$$

where $\bar{\cdot}$ denotes ℓ_2 normalization, and the pseudo-reward is the per-step reduction in that distance,

$$r_{t+k}^g = \kappa \left(d(e_{t+k}, g) - d(e_{t+k+1}, g)\right) ,$$

so that trajectories moving toward the goal in latent space accrue positive return.

The goal-conditioned head is a dueling architecture modulated by the goal through FiLM layers (Perez et al., 2017), and outputs a categorical distribution $p_\theta(\cdot \mid e_t, g)$ over a fixed support $\{z_1, \dots, z_N\}$ in the manner of C51 (Bellemare et al., 2017). It is trained with the distributional cross-entropy against the projected one-step target distribution $\Phi\hat{p}$ built from r_t^g:

$$\mathcal{L}_\theta^{\text{goal}} = -\sum_{i=1}^{N} (\Phi\hat{p})_i \log p_\theta(z_i \mid e_t, g) .$$

B.2.3 TRANSITION MODELS

The self-predictive objectives (BYOL and Barlow Twins) require a transition model to roll the representation forward. The online branch sees only the real observation at time t and imagines K steps ahead conditioned only on the action sequence, never observing the real future frames; the target branch encodes the actual future observations directly. The loss then compares the imagined and real representations at each step. Following Zhang et al. (2024), we consider three architectures and list their hyperparameters in Table 10. ATC, ST-DIM, and VAE-T use no transition model; their objectives compare representations of observations one step apart directly, so no rollout is performed.

Deterministic convolutional (`CD`). This model operates directly on the spatial convolutional feature map without flattening. The action is one-hot encoded and broadcast as additional spatial channels and concatenated onto the current feature map, which is passed through a convolution, normalization, and ReLU to a hidden width equal to the number of output channels (64 in our experiments), followed by a second convolution returning to the original channel count.[1]

Deterministic recurrent (`GD`). This model rolls the representation forward with a single-layer GRU cell (Chung et al., 2014), treating the representation itself as the recurrent state. The flattened representation is projected to the GRU hidden size and layer-normalized, and the action is embedded

[1] The implementation additionally supports a stack of inverted-residual blocks (Sandler et al., 2019) between the two convolutions and a zero-initialized residual connection around the model, following the SPR dynamics model of Schwarzer et al. (2021a); we leave both at their defaults, which disable them.

Table 10: Transition model architectures and their hyperparameters.

	CD	GD	GL
Hidden width	64	600	600
Residual blocks	0	—	—
GRU input size	—	250	600
Discrete latent	—	—	32×32
Latent merger	—	—	linear
KL balance β	—	—	0.95
Free nats	—	—	0
KL weight λ_{KL}	—	—	0.1
Renormalization	max	LayerNorm (no affine)	LayerNorm (no affine)
Rollout length `jumps`	10	10	10

through a learned lookup table; the GRU then steps forward, taking the action embedding as the timestep input and the projected representation as the recurrent hidden state. The model is purely deterministic, with no stochastic latent variable and no prior-posterior split.

Stochastic recurrent (`GL`). The richest of the three is a discrete-latent recurrent state-space model in the style of PlaNet (Hafner et al., 2019) and DreamerV2 (Hafner et al., 2022). A deterministic GRU hidden state h_t is updated at each step from the action and the previous stochastic latent. Two networks produce the categorical stochastic latent z_t: a posterior network conditioned on both h_t and the encoding of the real observation, used when encoding real trajectory data, and a prior network conditioned on h_t alone, used during imagination when future frames are unavailable. The deterministic and stochastic components are merged into the representation passed downstream.

The model is regularized by a KL term between posterior and prior, computed with straight-through one-hot categorical samples. Following Hafner et al. (2022), we balance the two directions of the divergence with a coefficient β:

$$\mathcal{L}^{\text{KL}} = (1-\beta)\,\mathrm{KL}(q(z_t \mid h_t, s_t) \,\|\, \mathrm{sg}[p(z_t \mid h_t)]) + \beta\,\mathrm{KL}(\mathrm{sg}[q(z_t \mid h_t, s_t)] \,\|\, p(z_t \mid h_t))\,,$$

with $\beta = 0.95$, which prevents the prior and posterior from collapsing onto one another too quickly in either direction.[2] For `GL` methods this term enters the total objective with weight $\lambda_{\text{KL}} = 0.1$.

B.2.4 Encoder architectures

The final field of each method name gives the encoder size. Sizes `M` and `L` are the same residual architecture at two scales, while `S` is a distinct, smaller, non-residual network; the size axis is therefore not a single scaling knob. All three produce a 7×7 spatial output on 84×84 inputs, so the sizes differ only in channel width and depth. An overview of each model is given in Table 11.

`S`. The three-layer convolutional stack of Mnih et al. (2015): convolutions with kernel sizes 8, 4, 3 and strides 4, 2, 1, widening to 32, 64, 64 channels, each followed by a ReLU. It has no residual connections and no normalization layers, and exposes no scale parameters, so it is a fixed architecture rather than a configurable size. The flattened representation is $64 \times 7 \times 7 = 3{,}136$ dimensional, from roughly 78k convolutional parameters.

`M` and `L`. These use the inverted-residual architecture of Espeholt et al. (2018) as adapted by Schwarzer et al. (2021a). Three stages each begin with a strided downsampling residual block followed by further blocks at the same width, where every block is an inverted bottleneck (Sandler et al., 2019) applying a 1×1 expansion, a depthwise 3×3 convolution, and a 1×1 projection, with batch normalization throughout. Three configuration values set the scale: a width multiplier on the base stage widths $[32, 64, 64]$, the number of residual blocks per stage, and the expansion ratio inside each block. `M` uses a multiplier of 1, 3 blocks per stage, and expansion ratio 2; `L` uses 1.5, 5, and 4 respectively. These compound, so `L` has roughly $6.5\times$ the parameters of `M` rather than the $1.5\times$ the width multiplier alone would suggest.

[2] The implementation additionally clips each term below at a free-nats floor, so that a term contributes no gradient once its KL falls below the threshold; the floor is left at its default of zero in every run reported here, where it is inactive.

Table 11: Encoder architectures. S and M produce identically shaped representations despite differing in parameter count by roughly $1.7\times$; the difference is depth and normalization, not width. L has both larger representation dimension and more parameters.

	S	M	L
Architecture	Nature-DQN	Impala/SPR	Impala/SPR
Residual blocks	0	9	15
Stage widths	32/64/64	32/64/64	48/96/96
Expansion ratio	—	2	4
Normalization	none	BatchNorm	BatchNorm
Output	$64 \times 7 \times 7$	$64 \times 7 \times 7$	$96 \times 7 \times 7$
Flattened dimension	3,136	3,136	4,704
Conv. parameters	$\sim$78k	$\sim$136k	$\sim$893k

C Data

In this section we describe the data we use in pretraining and probing in more detail. Section C.1 lists the games we evaluate on, and Section C.2 reports target statistics for each probing task.

Our data comes from RL Unplugged Atari (Gulcehre et al., 2020); whereas many previous works use the Atari Replay Dataset (Agarwal et al., 2020), the Atari Replay Dataset is no longer publicly available. The RLU dataset has been collected following the protocol described by Agarwal et al. (2020); thus while the data is not identical, we expect results to be broadly comparable, though we cannot verify this directly. The dataset is generated by recording the transitions of an online DQN agent trained for 50 million transitions. For each game, five independent runs are recorded and data is saved every 1 million steps, producing 50 checkpoints. Observations are 84×84 grayscale frames, stacked four deep. Rewards are sign-clipped to $\{-1, 0, +1\}$, following standard practice for Atari.

Pretraining data. Following Zhang et al. (2024), we pretrain on transitions from the first DQN run, using the first 500k transitions from each of the third, fourth, and fifth checkpoints, for a total pretraining dataset of 1.5 million transitions. As these transitions occur early in training, they correspond to data collected under a beginner-level policy.

Probing data. Probing data is drawn from the second DQN run, independent of the run used for pretraining data. We report results primarily using the last checkpoint, corresponding to a near-expert policy. We additionally perform ablations using a variety of early and intermediate checkpoints (see Appendix G.2) and find that our conclusions are robust to this choice. For each checkpoint, we collect 100k transitions and split them contiguously in time, using the first 20k for evaluation and the subsequent 80k for training. We note that the probing dataset is substantially smaller than the pretraining dataset; we analyze sensitivity to probing dataset size in Appendix G.3.

C.1 Game Selection

We run our experiments on 15 games from the ALE. Nine of these – Amidar, Assault, Asterix, Boxing, Demon Attack, Frostbite, Gopher, Krull, and Seaquest – follow Zhang et al. (2024), who study a similar setting. We additionally include Alien, Breakout, Kung Fu Master, Ms Pacman, Pong, and QBert, both to establish our results on a broader set of games and to test whether the findings of Zhang et al. extend to this additional set. We do not evaluate on more games due to compute constraints.

C.2 Target Statistics

For each of the probing tasks, we provide an overview of the target statistics of each game's evaluation dataset, by behavior-policy checkpoint.

Reward. We find that reward density is higher at the last checkpoint than the first in all 15 games, though the increase is not monotonic (see Table 12).

Terminal. We find that the number of positive terminal samples is never greater than 0.27%, indicating extreme class imbalance for this task (see Table 13). Despite this, the terminal probe remains competitive in every experiment in Section 5.

Next-action and inverse action. The targets for these two probes are the same, though the input differs. We find that while the marginal distribution over actions is near uniform in all games, the sequence is strongly auto-correlated, repeating the previous action 2.6–12$\times$ more often than chance. A flat marginal is not an absence of policy structure; it means only that no single action dominates (see Table 14).

n-step return. We find that mean return tends to increase with policy quality, similarly to the reward probe (see Table 15).

Temporal distance. Pairs are sampled log-uniformly in gap, so the marginal is close to uniform in all games at every checkpoint; the mild trend is that earlier checkpoints run shorter episodes and cannot supply the largest gaps, leaving the top buckets sparse (see Table 16).

Table 12: **Reward probe.** Percentage of the validation split with nonzero reward, by behavior-policy checkpoint. Δ is the change from the first to the last checkpoint.

Game	ck 1	ck 10	ck 25	ck 40	ck 50	Δ
alien	8.70	12.08	12.52	13.46	14.08	+5.38
amidar	2.88	4.28	5.10	5.23	5.42	+2.54
assault	3.49	5.87	6.53	6.77	6.66	+3.17
asterix	5.11	5.71	6.04	5.84	5.58	+0.46
boxing	3.85	9.13	10.79	11.65	9.55	+5.70
breakout	1.90	3.16	3.00	3.04	3.10	+1.20
demon_attack	1.99	4.27	4.79	4.78	4.78	+2.79
frostbite	3.98	9.88	9.84	11.23	5.88	+1.91
gopher	3.02	6.16	7.85	7.72	7.28	+4.26
krull	13.05	38.36	42.02	44.11	41.98	+28.93
kung_fu_master	4.08	4.79	4.76	4.09	4.52	+0.43
ms_pacman	13.07	15.33	14.50	15.25	15.75	+2.68
pong	0.90	0.91	0.87	0.90	0.97	+0.08
qbert	3.58	11.58	13.64	13.88	13.58	+10.00
seaquest	1.52	2.99	6.35	7.61	7.46	+5.94

Table 13: **Terminal probe.** Count of terminal samples in the 20k-row validation split, by checkpoint. The last column is the positive rate at whichever checkpoint supplies the most.

Game	ck 1	ck 10	ck 25	ck 40	ck 50	best rate
alien	30	24	21	17	16	0.150%
amidar	22	11	8	6	7	0.110%
assault	24	15	19	19	20	0.120%
asterix	41	23	20	23	24	0.205%
boxing	12	18	22	23	20	0.115%
breakout	53	15	14	18	15	0.265%
demon_attack	20	7	8	5	5	0.100%
frostbite	35	27	27	23	31	0.175%
gopher	18	12	9	10	9	0.090%
krull	14	15	16	15	14	0.080%
kung_fu_master	9	7	6	8	6	0.045%
ms_pacman	24	19	17	17	15	0.120%
pong	7	6	6	7	7	0.035%
qbert	30	18	12	19	14	0.150%
seaquest	20	13	9	13	14	0.100%

Table 14: **Next-action and inverse action probes.** Normalized action entropy $H/H_{\max}$ by checkpoint, with action-space size $|\mathcal{A}|$ and the lag-1 repeat rate at the final checkpoint as a multiple of chance.

Game	$\|\mathcal{A}\|$	ck 1	ck 10	ck 25	ck 40	ck 50	$P(\text{rep})$
alien	18	0.981	0.990	0.993	0.993	0.995	10.3×
amidar	10	0.975	0.975	0.971	0.965	0.949	6.4×
assault	7	0.954	0.949	0.932	0.934	0.950	3.3×
asterix	9	0.922	0.986	0.996	0.997	0.994	5.7×
boxing	18	0.964	0.962	0.956	0.965	0.961	5.4×
breakout	4	0.926	0.985	0.992	0.986	0.984	2.9×
demon_attack	6	0.983	0.998	0.989	0.980	0.985	2.8×
frostbite	18	0.992	0.988	0.991	0.990	0.986	12.0×
gopher	8	0.952	0.985	0.963	0.976	0.977	4.2×
krull	18	0.966	0.982	0.984	0.988	0.985	8.2×
kung_fu_master	14	0.945	0.989	0.989	0.990	0.994	8.1×
ms_pacman	9	0.991	0.998	0.996	0.994	0.990	5.8×
pong	6	0.986	0.989	0.996	0.998	0.994	2.6×
qbert	6	0.990	0.995	0.998	0.997	0.999	4.0×
seaquest	18	0.979	0.992	0.987	0.991	0.993	9.2×

Table 15: **n-step return probe.** Mean ± s.d. of the target by checkpoint (n=10, γ=0.99, sign-clipped rewards), and the fraction of exactly-zero returns at the last checkpoint.

Game	ck 1	ck 10	ck 25	ck 40	ck 50	G=0
alien	0.83 ±1.00	1.16 ±1.14	1.20 ±1.13	1.29 ±1.13	1.35 ±1.12	32%
amidar	0.28 ±0.56	0.41 ±0.63	0.49 ±0.67	0.50 ±0.67	0.52 ±0.68	58%
assault	0.33 ±0.49	0.56 ±0.60	0.62 ±0.62	0.65 ±0.64	0.63 ±0.64	45%
asterix	0.47 ±0.99	0.54 ±0.93	0.58 ±0.94	0.55 ±0.94	0.53 ±0.94	68%
boxing	0.02 ±0.65	0.55 ±1.11	0.77 ±1.18	0.81 ±1.22	0.64 ±1.11	43%
breakout	0.18 ±0.38	0.30 ±0.44	0.29 ±0.47	0.29 ±0.47	0.30 ±0.45	69%
demon_attack	0.19 ±0.40	0.41 ±0.57	0.46 ±0.59	0.46 ±0.61	0.46 ±0.59	58%
frostbite	0.38 ±0.59	0.94 ±1.90	0.94 ±1.92	1.07 ±2.06	0.56 ±1.30	67%
gopher	0.29 ±0.65	0.59 ±0.96	0.75 ±1.01	0.74 ±1.00	0.70 ±0.99	59%
krull	1.25 ±2.14	3.67 ±3.32	4.02 ±3.41	4.22 ±3.42	4.01 ±3.33	31%
kung_fu_master	0.39 ±0.65	0.46 ±0.70	0.46 ±0.69	0.39 ±0.66	0.43 ±0.69	66%
ms_pacman	1.25 ±1.27	1.46 ±1.26	1.39 ±1.26	1.46 ±1.25	1.51 ±1.25	34%
pong	−0.05 ±0.28	0.05 ±0.28	0.04 ±0.28	0.05 ±0.28	0.06 ±0.29	90%
qbert	0.34 ±0.71	1.11 ±1.88	1.30 ±2.06	1.33 ±2.05	1.30 ±2.04	40%
seaquest	0.15 ±0.35	0.29 ±0.46	0.61 ±0.75	0.73 ±0.77	0.71 ±0.78	48%

D More on Proxies

In this section, we provide additional information about each of the proxies. Section D.1 explains implementation details, Section D.2 examines whether proxies are biased to the representation dimension, Section D.3 explains the technical details of comparing the proxies (see Section 5.5), and Section D.4 discusses boundary cases of NFoO.

D.1 Probe Fitting

Each linear probing task fits into one of three categories: binary classification, multi-class classification, and regression. Probe hyperparameters are consistent across all games and model selection scenarios. The number of iterations or epochs was chosen to be large enough for convergence of the probe's loss on its training dataset. Effective rank requires no fitting and is handled separately below.

Binary probes (reward, terminal). We use Cyanure's logistic classifier with an ℓ_2 penalty, $\lambda_1 = 4 \times 10^{-9}$, `tol` $= 10^{-5}$, at most 300 iterations, and no intercept. Features are centered and row-ℓ_2-normalized by Cyanure's `preprocess` (`centering=True, normalize=True,`

Table 16: **Temporal-distance probe.** Normalized class entropy $H/H_{\max}$ over the 10 octave buckets, by checkpoint. Δ is the change from the first to the last checkpoint.

Game	ck 1	ck 10	ck 25	ck 40	ck 50	Δ
alien	0.962	0.970	0.978	0.987	0.988	+0.026
amidar	0.977	0.994	0.997	0.998	0.998	+0.020
assault	0.972	0.990	0.983	0.981	0.981	+0.009
asterix	0.940	0.975	0.981	0.977	0.973	+0.033
boxing	0.995	0.983	0.976	0.972	0.981	-0.014
breakout	0.919	0.991	0.992	0.986	0.989	+0.070
demon_attack	0.984	0.997	0.997	0.998	0.998	+0.014
frostbite	0.946	0.964	0.966	0.973	0.954	+0.008
gopher	0.985	0.993	0.996	0.996	0.996	+0.011
krull	0.992	0.990	0.988	0.990	0.991	-0.001
kung_fu_master	0.997	0.998	0.998	0.997	0.998	+0.001
ms_pacman	0.969	0.982	0.986	0.987	0.989	+0.019
pong	0.998	0.998	0.998	0.998	0.997	0.000
qbert	0.964	0.985	0.993	0.982	0.991	+0.027
seaquest	0.980	0.991	0.996	0.992	0.990	+0.010

`columns=False`) on both train and validation splits. The reported metric is scikit-learn's binary F_1, which we prefer to accuracy given the class imbalance documented in Appendix C.2.

Multi-class probes (next action, temporal distance, inverse action). A single `nn.Linear` head trained by SGD (learning rate 0.2, momentum 0.9, weight decay 10^{-6}) for 12 epochs at batch size 256, with `StepLR` decaying the learning rate by a factor of 0.1 at epoch 10, gradient-norm clipping at 0.1, and a softmax focal loss ($\gamma_{\mathrm{f}} = 2$). The reported metric is scikit-learn's weighted F_1 (`zero_division= 0`). Note that the temporal distance and inverse action probes are the only probes requiring two input representations. The inverse-action probe concatenates them as $[y, y']$; the temporal-distance probe additionally appends the elementwise squared difference, $[y, y', (y' - y)^2]$, since temporal distance grows with $\|y' - y\|$, which no linear functional of the raw concatenation can express[3].

Regression probes (n-step return). Regression-based probes use scikit-learn's ridge regression with $\alpha = 1$, on preprocessed features as in the binary case, scored by R^2 on the held-out split.

Effective rank. Effective rank has no head and no preprocessing: it is computed directly on the raw feature matrix, as defined in Section 3.1. Unlike the other proxies, it is evaluated on 50k transitions from the pretraining dataset, following Kumar et al. (2021).

D.1.1 Temporal Distance Bins

Temporal distance asks whether the representation encodes relational structure, namely how far apart in time two states are. We sample a pair of representations $(\phi(s_t), \phi(s_{t+n}))$ from within one episode and predict which of K buckets n falls into. Buckets are logarithmically spaced: class k covers $n \in [2^k, 2^{k+1})$, and $K = \lfloor \log_2 n_{\max} \rfloor + 1$ with $n_{\max} = 1000$, the top class being truncated at $n_{\max}$.

D.2 Does representation dimension bias the proxies?

Fourteen of the sixteen pretraining methods, covering all S and M models, produce representations of dimension $d = 3136$, while the two L methods produce $d = 4704$. Under both Rainbow and PPO, the two L methods are also the strongest of the sixteen pretraining methods after finetuning. A proxy that ranks them first may be responding to the quality of their representations or simply to their larger size, and on the full pool the two explanations cannot be separated. This raises the question of whether the proxy results are biased by representation dimension.

[3]Early in experimentation we found that a linear head on the raw concatenation scored near chance, while adding the squared-difference term improved results substantially.

Table 17: **Proxy and finetuning scores of random frozen encoders at the two representation dimensions of our pool.** Neither encoder has learned anything, so the ratio is what each measure awards for the larger representation alone. A ratio near 1 means the measure gains nothing from the extra dimensions. For every measure, the ratio is taken within each game and the table reports the IQM of the 15 per-game ratios. The two score columns are averaged over all 15 games and are shown for reference, so the ratio need not equal their quotient. Proxy and finetuning scores both average seeds 1–5; for these encoders each seed is a different random initialization. Finetuning uses raw game scores, shifted per game so the pooled minimum across both encoder sizes is zero (raw scores can be negative). Intervals are 95% bootstrap intervals over seeds.

Measure	$d = 3136$	$d = 4704$	ratio	95% CI
Terminal	0.057	0.052	1.00×	0.97–1.01
Reward	0.458	0.484	1.11×	1.08–1.13
n-step return	0.353	0.354	1.01×	1.00–1.02
Next action	0.128	0.128	1.03×	1.01–1.05
Inverse action	0.131	0.144	1.08×	1.05–1.11
Temporal distance	0.102	0.097	0.98×	0.94–1.03
Effective rank	1915	2764	**1.45**×	1.44–1.45
Finetune, Rainbow (100k)	—	—	1.07×	0.95–1.26
Finetune, PPO (10M)	—	—	1.01×	0.99–1.04

To isolate the effect of dimension, we compare random frozen encoders at the two dimensions (Table 17). For each measure, we take the ratio of the larger encoder's score to the smaller one's within each game and report the IQM over all 15 games. Finetuning shows no significant difference: the ratio is 1.07× under Rainbow (95% bootstrap interval 0.95–1.26) and 1.01× under PPO (0.99–1.04).. The six supervised proxies also stay close to 1, with ratios between 0.98× and 1.11×. Effective rank does not: it rises by 1.45×, well outside both finetuning intervals, because the number of significant singular values grows with the number of features whether or not those features carry information. On the full pool, part of effective rank's preference for the L methods therefore comes from their size alone, which finetuning does not reward.

A fair comparison therefore has to hold dimension fixed, which means setting the two L methods aside. A second bias has to go as well. As discussed in Section 5.1, effective rank systematically over-ranks the two ST-DIM methods (`ST-DIM-S`, `ST-DIM+I-S`): they have the highest effective rank among the S and M models, yet they finetune poorly under Rainbow. Left in the pool, they would dominate effective rank's results and hide how it behaves on every other method. We set them aside too, which leaves twelve pretraining methods that all share $d = 3136$. At the game level, where every (method, checkpoint) pair is a separate candidate, seven of these methods have all ten checkpoints finetuned, which gives 70 representations per game. The four tables in this section repeat the calculations in Section 5.1 and 5.3 on this reduced pool.

At the method level (Tables 18 and 19), the results follow the same split as in Section 5.1. Under Rainbow, the value probes correlate best with finetuning: n-step return reaches $\rho = +0.96$ and reward $+0.92$, and both select the oracle, `BT-Rand+I-GL-M`. Under PPO, the action and temporal probes correlate best: next action ($+0.89$), temporal distance ($+0.85$) and inverse action ($+0.83$) all select the oracle, `BYOL+GI-GD-M`, while reward falls to $+0.52$ ($p = 0.08$). Effective rank reaches $+0.92$ under Rainbow and $+0.69$ under PPO. Every proxy's top-1 pick has an NFoO of at least 0.84, compared with 0.41 (Rainbow) and 0.49 (PPO) for a random pick.

At the game level (Tables 20 and 21), each (method, checkpoint) pair is a separate candidate, as in Appendix A.2, but the pool shrinks from 100 to 70 representations. The n-step return is the strongest proxy in this setting: its median ρ over the 15 games is 0.84, its lowest is 0.41, its top-1 pick is never below the random baseline, and its median NFoO is 0.86/0.90/0.95 at top-1/2/3. Reward is next (median $\rho = 0.75$, top-1 NFoO 0.85, never below random), followed by effective rank (median $\rho = 0.70$, top-1 NFoO 0.78, below random on 3 games). Terminal is the weakest: its median ρ of 0.61 is tied with temporal distance for the lowest, but its top-1 pick falls below random on 6 of the 15 games (temporal distance: 1), and its median top-1 NFoO, 0.57, is the lowest of all seven proxies.

Table 18: **Spearman ρ over the twelve pretraining methods of equal dimension.** As in Section 5.1, but restricted to the S and M methods at $d = 3136$, excluding `ST-DIM-S` and `ST-DIM+I-S`.

		Rainbow (100k)		PPO (10M)	
Probe	Metric	ρ	p	ρ	p
n-step return	R^2	$+0.958$	< 0.001	$+0.727$	0.007
Reward	Binary F1	$+0.923$	< 0.001	$+0.524$	0.080
Effective rank	srank	$+0.923$	< 0.001	$+0.685$	0.014
Terminal	Binary F1	$+0.860$	< 0.001	$+0.594$	0.042
Next action	Weighted F1	$+0.818$	0.001	$+0.888$	< 0.001
Inverse action	Weighted F1	$+0.769$	0.003	$+0.825$	< 0.001
Temporal distance	Weighted F1	$+0.727$	0.007	$+0.846$	< 0.001

Table 19: NFoO **over the twelve pretraining methods of equal dimension, without ST-DIM.** Same pool as in Table 18. The oracle is `BT-Rand+I-GL-M` under Rainbow and `BYOL+GI-GD-M` under PPO. *Random* is the expected NFoO from selecting a single method uniformly at random and is the reference for the top-1 column.

		Rainbow (100k)			PPO (10M)		
Probe	Selected method	top-1	top-2	top-3	top-1	top-2	top-3
Reward	`BT-Rand+I-GL-M`	1.000	1.000	1.000	0.856	0.856	0.856
n-step return	`BT-Rand+I-GL-M`	1.000	1.000	1.000	0.856	1.000	1.000
Effective rank	`BT-Rand+I-GL-M`	1.000	1.000	1.000	0.856	1.000	1.000
Terminal	`BYOL+GI-GD-M`	0.839	1.000	1.000	1.000	1.000	1.000
Next action	`BYOL+GI-GD-M`	0.839	0.839	1.000	1.000	1.000	1.000
Inverse action	`BYOL+GI-GD-M`	0.839	0.839	0.839	1.000	1.000	1.000
Temporal distance	`BYOL+GI-GD-M`	0.839	1.000	1.000	1.000	1.000	1.000
Random	—		0.408			0.492	

The proxies are therefore not an artifact of representation dimension. On the full pool, which includes the L methods, every proxy correlates significantly with finetuning under both learners and every top-1 pick beats a random one (Section 5.1). On the equal-dimension pool, every correlation stays positive, all but one remain significant, and every top-1 pick reaches an NFoO of at least 0.84. The one dimension effect we find is in effective rank. On untrained encoders, where finetuning barely changes with dimension, effective rank still rises by $1.45\times$. When two representations finetune about equally well, effective rank will therefore favor the one with more features.

D.3 Proxy Comparisons

In this section, we give a comparison of different proxies. *Reward* and *n-step return* both predict a function of the reward signal, and *next action* and *inverse action* both predict which action was taken, *terminal* predicts whether an observation ends an episode, *temporal distance* predicts how far apart two observations are in time, and *effective rank* measures the encoder's features directly. This appendix measures how far that overlap in objective carries through to the rankings the proxies produce.

Two sets of pretrained encoders. We run the comparison twice, on two sets of encoders. The first, from Section 5.1, is the 16 pretraining methods at their final checkpoints: 16 encoders per game, 240 in all. The second, from Section 5.3, adds the intermediate checkpoints, taking 10 points along the pretraining run of each of 10 methods: 100 encoders per game, 1500 in all.

Every encoder carries eight numbers, and every comparison in this appendix is between two of them. Seven are the scores from each proxy, and the eighth is the encoder's downstream score, its mean human-normalized return over 10 seeds of Rainbow DQN finetuning at 100K environment steps.

Measuring agreement. Within a game, any two of those eight quantities induce two orderings of the encoders, and we take the Spearman ρ between them. Normalized fraction of oracle cannot

Table 20: Per-game Spearman ρ over the 70-representation pool. As Table 4, but without the two L methods and ST-DIM-S, each at 10 pretraining checkpoints. The last row is the median over the 15 games.

Game	Terminal	Reward	n-step return	Next action	Inverse action	Temporal distance	Effective rank
alien	0.66	0.75	0.88	0.89	0.89	0.81	0.69
ms_pacman	0.72	0.72	0.84	0.83	0.83	0.70	0.81
assault	0.68	0.75	0.81	0.94	0.76	0.67	0.84
seaquest	0.73	0.85	0.90	0.65	0.72	0.31	0.84
qbert	0.62	0.59	0.89	0.53	0.79	0.83	0.68
breakout	0.77	0.84	0.86	0.54	0.29	0.76	0.84
gopher	0.35	0.86	0.85	0.84	0.79	0.44	0.70
demon_attack	0.42	0.78	0.69	0.79	0.72	0.73	0.63
pong	0.19	0.82	0.89	0.70	0.54	0.61	0.72
boxing	0.71	0.84	0.90	0.47	0.64	-0.18	0.87
asterix	0.61	0.63	0.81	0.52	0.44	0.61	0.55
amidar	-0.15	0.73	0.77	0.57	0.57	0.79	0.68
krull	0.22	0.73	0.77	0.57	0.69	0.06	0.66
kung_fu_master	0.00	0.49	0.41	0.75	0.67	0.46	0.52
frostbite	0.52	0.52	0.73	0.20	0.28	0.22	0.72
Median	**0.61**	**0.75**	**0.84**	**0.65**	**0.69**	**0.61**	**0.70**

Table 21: Per-game normalized fraction of oracle (NFoO) over the 70-representation pool. As Table 5, but over the pool of Table 20. Each cell gives top-1/top-2/top-3, with leading zeros dropped. *Random* is the expected top-1 NFoO from drawing one representation uniformly at random. The last row is the median over the 15 games.

Game	Terminal	Reward	n-step return	Next action	Inverse action	Temporal distance	Effective rank	Random
alien	.26/.69/.69	.86/.86/.86	.86/.86/.86	.61/.86/.86	.86/.86/.86	.76/.96/1	.60/.60/.60	.47
ms_pacman	.35/.97/.97	.96/1/1	.96/1/1	.91/.97/.97	.69/.91/.91	.66/.66/.66	.97/.97/.97	.48
assault	.57/.57/.57	.76/.76/.76	.72/.76/.76	.77/.84/.84	.52/.52/.52	.52/.57/.57	.57/.57/.77	.38
seaquest	.79/.79/.79	.85/.85/.85	.89/.89/.89	.73/.80/.80	.73/.80/.80	.78/1/1	.85/.85/.85	.48
qbert	.83/.83/.83	.77/.77/1	.82/1/1	.88/.88/1	.86/.92/.92	.82/.92/1	.87/.87/.88	.56
breakout	.20/.26/.41	1/1/1	1/1/1	.69/.69/.69	.69/.69/.69	.69/.69/.69	.69/.69/.83	.25
gopher	.67/.70/.72	.85/.85/.85	.75/.85/.85	.77/.77/.77	.73/.85/.85	.73/.73/.73	.41/.44/.77	.50
demon_attack	.78/.89/.93	.28/.34/.36	.34/.44/.45	.61/.81/.81	.44/.44/.44	.36/.37/.43	.06/.07/.11	.24
pong	.58/.67/.71	.66/.88/.95	.88/.90/.95	.46/.46/.49	.71/.71/.71	.71/.71/.71	.78/.78/.78	.47
boxing	.79/.88/.88	.77/.80/.80	.92/1/1	.73/.75/.75	.59/.71/.71	.71/.71/.71	.92/1/1	.58
asterix	.28/.49/.49	.87/.87/.99	.90/.90/.99	.32/.35/.35	.32/.35/.35	.99/.99/.99	.20/.20/.20	.29
amidar	.23/.23/.55	.57/.71/.83	.89/.89/.92	.41/.41/.46	.41/.41/.43	.89/.89/.92	.83/.83/.83	.52
krull	.84/.84/.84	.93/.93/.93	.76/1/1	.76/.77/.77	.77/.88/.88	0/.17/.26	.82/.82/.82	.54
kung_fu_master	.51/.68/.75	.85/.96/.96	.63/.96/.96	.67/.82/.96	.96/.96/.96	.96/.96/.96	.82/.82/.82	.51
frostbite	.36/.94/.94	.55/.71/.71	.78/.94/.94	.70/.70/.70	.81/.94/.94	.81/.94/.94	.56/1/1	.51
Median	**.57/.70/.75**	**.85/.85/.86**	**.86/.90/.95**	**.70/.77/.77**	**.71/.80/.80**	**.73/.73/.73**	**.78/.82/.82**	**.48**

serve here. It scores a proxy only by the encoders it ranks at the top, so comparing two proxies with it reduces to asking whether their top picks coincide, and says nothing about how the rest of their orderings agree. The 15 per-game values are averaged.

Correlating two proxies directly would not answer the question. If each of them ranks encoders roughly the way finetuning does, then the two must also rank encoders roughly like each other.

We therefore subtract from each proxy the part of its ranking that the downstream score already accounts for, and correlate what is left. For proxies A and B with downstream score f, writing ρ_{AB} for their raw agreement and ρ_{Af}, ρ_{Bf} for each proxy's correlation with f,

$$\rho_{AB\cdot f} \;=\; \frac{\rho_{AB} - \rho_{Af}\,\rho_{Bf}}{\sqrt{(1-\rho_{Af}^2)(1-\rho_{Bf}^2)}}.$$

What remains is agreement about *where the two proxies are wrong*. A high value means they are right and wrong about the same encoders; a value near zero means their errors are unrelated. We call this the *shared error*.

Different encoders, same conclusion. The two sets pose different problems: the second asks a proxy to order the per game representations, not just one method against another. Individual cells of Table 22 and Table 23 differ by up to 0.21, but the two tables give the same answer: in both, the same two pairs are the only ones above 0.52, so everything below holds for either.

Table 22: **Shared error,** 16 **pretraining methods.** Agreement between two proxies once each one's correlation with the downstream score has been removed. The matrix is symmetric, so only the lower triangle is shown and columns are indexed by row number. Rows are ordered so that each of the two close pairs is adjacent, identically in both tables; bold marks those two pairs.

Proxy	(1)	(2)	(3)	(4)	(5)	(6)
(1) n-step return						
(2) Reward	**0.632**					
(3) Next action	0.303	0.206				
(4) Inverse action	0.259	0.197	**0.798**			
(5) Effective rank	0.366	0.217	0.482	0.374		
(6) Terminal	0.304	0.309	0.143	0.157	0.165	
(7) Temporal distance	0.414	0.241	0.239	0.123	0.391	0.232

Table 23: **Shared error,** 100 **representations.** As Table 22, but ranking 10 checkpoints from each of 10 methods rather than 16 methods at their final checkpoints.

Proxy	(1)	(2)	(3)	(4)	(5)	(6)
(1) n-step return						
(2) Reward	**0.628**					
(3) Next action	0.512	0.300				
(4) Inverse action	0.465	0.353	**0.745**			
(5) Effective rank	0.459	0.328	0.393	0.259		
(6) Terminal	0.274	0.263	0.221	0.246	0.174	
(7) Temporal distance	0.299	0.107	0.381	0.235	0.374	0.115

Two pairs are alike; the rest are not. Removing the downstream component cuts mean agreement from 0.55 and 0.57 to 0.31 and 0.34 on the two pools, and only two pairs stay high: *next action* with *inverse action* at 0.798 and 0.745, and *reward* with n*-step return* at 0.632 and 0.628. In both, one proxy is close to a special case of the other. *Reward* predicts whether the immediate reward is nonzero, a coarsened version of what n*-step return* regresses at $n = 1$, and *next action* is *inverse action* shown only the first of its two observations. Each pair recovers related information from overlapping inputs. Nothing else reaches 0.52; the rest average 0.27 and 0.30, down to 0.123 for *inverse action* against *temporal distance* on the 16 methods and 0.107 for *reward* against *temporal distance* on the 100 representations. Sharing part of a training signal is thus not enough on its own to make two proxies alike: outside these pairs they are wrong about different encoders, and appear to measure different properties of the representation.

D.4 Boundary Cases

The evaluation of proxies in certain boundary cases requires particular care. We specify our treatment of these cases below.

Ties. When a proxy assigns the same score to several candidates, a top-k rule may not uniquely determine which candidates are selected. We score such a pool by taking the expectation over uniformly random tie-breaking. Write $\tilde{y}$ for a candidate's normalized fraction of oracle, and let t be the proxy score of the k-th ranked candidate. Every candidate with proxy score strictly above t is selected. Let k' be the number of these candidates and c the largest $\tilde{y}$ among them, with $c = -\infty$ if $k' = 0$.

The remaining $r = k - k'$ slots are filled by a uniformly random subset of the m candidates whose proxy score equals t. Writing $\tilde{y}_{(1)} \leq \cdots \leq \tilde{y}_{(m)}$ for their normalized fraction of oracle scores, we define the NFoO_k of the proxy on this pool as

$$\mathrm{NFoO}_k = \binom{m}{r}^{-1} \sum_{i=r}^{m} \binom{i-1}{r-1} \max\big(c,\ \tilde{y}_{(i)}\big).$$

When the boundary group is selected in full ($m = r$), this expression reduces to the largest $\tilde{y}$ among the top k candidates. If the proxy is constant on the entire pool, then $k' = 0$, $r = k$, and $c = -\infty$, and the resulting score equals the random-selection baseline for that budget.

Spearman's ρ already averages tied ranks. A proxy constant on the whole pool induces no ranking at all and is scored $\rho = 0$.

Degenerate downstream-score ranges. NFoO is undefined when all candidates have the same downstream score. This case should be reported separately and excluded from NFoO averages, because downstream performance does not distinguish among candidates.

E FINETUNING PROTOCOL

This section describes how pretrained encoders are finetuned and evaluated. Section E.1 covers the Rainbow agent, Section E.2 the PPO agent, Section E.3 the learning-rate protocol, and Section E.4 evaluation and score normalization.

E.1 RAINBOW AT 100K

Every finetuning run loads a pretrained encoder, freezes it, and trains a fresh value head for 100,000 environment steps. All auxiliary objectives used during pretraining are switched off, so the finetuning loss is the distributional TD loss alone. Table 24 gives the full configuration; only the head learning rate varies across runs (see Appendix E.3).

We refer to the agent as Rainbow, but it is Rainbow without double DQN enabled: distributional ($C51$), dueling, prioritized replay, multi-step returns and noisy networks are all enabled, while double Q-learning is not, so the bootstrap action is chosen by the target network. This configuration is inherited from the SGI codebase (Schwarzer et al., 2021b) by way of Zhang et al. (2024).

E.2 PPO AT 10M

The PPO arm exists to test whether the ranking a proxy predicts depends on the downstream learner. It uses the same frozen encoders and the same 15 games, but a different algorithm, a $100\times$ larger interaction budget, and freshly initialized policy and value heads. Table 25 gives the configuration.

E.3 LEARNING-RATE PROTOCOL

A proxy's ability to anticipate downstream performance is only meaningful relative to a reasonably well-tuned learner: if the downstream RL agent is poorly configured, its performance may be dominated by that misconfiguration rather than by the quality of the underlying representation, in which case no proxy could be expected to predict it. We therefore select the Rainbow learning rate by sweeping $\{3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$ for all 16 methods.

Table 26 gives the winning learning rate in each (game, method) pair. No single learning rate is optimal across all methods; rather, the best learning rate depends on encoder size. The largest encoder prefers the smallest learning rate (3×10^{-5}) in 63% of cells against 26% under the smallest encoder. In contrast, the smallest encoder prefers the largest learning rate (3×10^{-4}) in 39% of cells compared to only 3% under the largest encoder. Fixing one global learning rate would therefore be biased: it would penalize one end of the encoder spectrum in a way that our proxies cannot see.

The learning rate matters at two scales. Within a cell the impact may be large: the gap between the best and worst rate has a median of 0.090 HNS and reaches 2.505 (Table 27), compared with a median of 0.54 HNS for the entire choice of pretraining method on the same cells (mean 1.54,

Table 24: Rainbow finetuning configuration. Values are the same for every method, game and checkpoint unless noted.

	Setting	Value
Environment	Interaction budget	10^5 environment steps (Atari 100k)
	Games	15
	Observation	84×84 grayscale, 4-frame stack
	Action set	minimal per game ($4 \leq \lvert\mathcal{A}\rvert \leq 18$)
	Sticky actions	$p = 0.25$, as in the pretraining data
	Training augmentation	random shift + intensity
	Evaluation augmentation	none
Agent	Encoder	frozen, loaded from pretraining
	Distributional atoms	51
	Value support $[V_{\min}, V_{\max}]$	$[-10, 10]$
	Dueling head	yes
	Head width	512
	Double Q	no
Optimisation	Optimizer	Adam, $\varepsilon = 1.5 \times 10^{-4}$, weight decay 10^{-6}
	Batch size	64
	Replay ratio	64
	Learning rate	$\{3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$
	Gradient-norm clip	10
	Discount γ	0.99
	n-step return	10
	Steps before learning	2,000
Replay/target	Prioritized replay	yes, $\alpha = 0.5$
	β anneal	over 10^5 steps
	Target update	every step, $\tau = 0.01$
	Value-head reset	off
	ε-greedy	$1 \to 0$ over 2,001 steps; $\varepsilon_{\text{eval}} = 10^{-3}$
Eval	Parallel environments	100
	Episodes per evaluation	up to 100
	Decorrelation steps	0

reaching 5.54 in `boxing`). In the worst cells, an unlucky learning rate therefore costs more than a typical choice of method. These large gaps are concentrated in a minority of cells: the mean best–worst gap (0.272 HNS) is three times the median, and the default rate is within 0.003 HNS of the best in the median cell. In aggregate, the ranking is robust: ranking the methods under any single fixed learning rate reproduces the swept ranking at $\rho \geq 0.93$. Only the largest rate, 3×10^{-4}, changes the top pick, to `BT-Rand+I-GL-M`.

The sweep therefore matters for the absolute performance of individual runs but has little effect on the rank correlations and model-selection conclusions we report. Our protocol reflects this in two ways. First, learning-rate selection mirrors our two application granularities. The cross-method experiments (Sections 5.1 and 5.2) compare methods by aggregate performance, so each method is finetuned at the single rate maximizing its IQM across the 15 games (the *Aggregate* row of Table 26). The per-game and early-stopping experiments (Sections 5.3 and 5.4) operate within a single game, so each (game, method) pair uses its own winner from the body of Table 26, selected at the final checkpoint and reused for all earlier ones. Second, we do not repeat the sweep for PPO due to its computational cost; we take the Rainbow result as suggestive, though not verified, evidence that the same insensitivity holds there.

E.4 Evaluation and Normalization

At the end of finetuning, both Rainbow and PPO agents are evaluated in the same way: across 100 parallel environments, each with its own seed, collecting up to 100 complete episodes. Evaluation is greedy up to $\varepsilon = 10^{-3}$ for Rainbow and samples the policy for PPO, and no augmentation is applied. The recorded score is the mean undiscounted game score over the episodes of the final evaluation.

Table 25: PPO finetuning configuration. Values are identical for every method, game and seed.

	Setting	Value
Environment	Interaction budget	10^7 environment steps
	Games	15
	Observation	84×84 grayscale, 4-frame stack
	Action set	minimal per game ($4 \leq \lvert\mathcal{A}\rvert \leq 18$)
	Sticky actions	$p = 0.25$, as in the pretraining data
	Augmentation	none
Agent	Encoder	frozen, loaded from pretraining
	Representation	flattened feature map, LayerNorm without affine
	Policy head	$\texttt{Linear}(d, 512)$–ReLU–$\texttt{Linear}(512, \lvert\mathcal{A}\rvert)$
	Value head	$\texttt{Linear}(d, 512)$–ReLU–$\texttt{Linear}(512, 1)$
	Trainable parameters	3.22M (S/M), 4.83M (L) at $\lvert\mathcal{A}\rvert = 18$
Rollout	Parallel environments	8
	Rollout length	128 steps
	Steps per iteration	1,024
	Iterations	$\approx$ 9,766
	Epochs per iteration	4
	Minibatches per epoch	4
Objective	Optimizer	Adam, default $\varepsilon = 10^{-8}$, no weight decay
	Learning rate	2.5×10^{-4}, annealed linearly to 0
	Discount γ	0.99
	GAE λ	0.95
	Ratio clip	0.1
	Value loss coefficient	0.5
	Entropy bonus	0.01
	Advantage normalization	yes
	Gradient-norm clip	1.0
Eval.	Parallel environments	100
	Episodes per evaluation	up to 100 episodes over 100 parallel environments

Since raw game scores are not comparable across games, we report the standard human-normalized score, $\frac{\text{agent score} - \text{random score}}{\text{human score} - \text{random score}}$. The values of the human and random scores are taken from Wang et al. (2016). When we report scores at a per-game level, scores are averaged over 10 seeds. When scores are reported across games, we compute the IQM (Agarwal et al., 2021) over all seed-level scores pooled across games.

F Compute

The premise of this work is that finetuning is expensive and a proxy is cheap. This section puts numbers on both, drawn from the project's own job records, and closes with the ratio between them.

Hardware. Every run requested a single GPU on a shared academic cluster spanning roughly a dozen GPU models, with V100-SXM2, A100-SXM4, L40S, and V100-PCIE accounting for 76–82% of runs at every stage. All costs reported here are medians over this heterogeneous fleet. Proxy and finetuning runs drew from the same pool of nodes, so the comparison between them is not distorted by hardware.

Run counts. The finetuning workload follows from the experimental design (Section 4) and is given in Table 28. The two Rainbow arms overlap wherever a checkpoint-sweep run coincides with a cross-method run at the selected learning rate; such runs are counted once. Pretraining and probing instead cover the full pool of $16 \times 15 = 240$ (method, game) pairs, each contributing one pretraining run and one probing job over that pair's ten checkpoints.

Cost per run. Table 29 gives the median cost of a single run at each stage. Cost grows steeply with encoder size at every stage, so totals are computed by weighting each run set by its own encoder mix rather than by a single global average.

Table 26: Selected Rainbow learning rate for each (game, method): $-$ is 3×10^{-5}, $\cdot$ is 10^{-4} (the default), $+$ is 3×10^{-4}. The sweep is run on the final pretraining checkpoint with 10 seeds per arm. The *Aggregate* row gives the rate maximizing each method's IQM'd score over the pooled 15-game $\times$ 10-seed set, which need not agree with any per-game winner. The final three rows count each rate's wins per method and sum to 15 games per column.

Game	BT-0.7+I-GL-L	BT-Rand+I-GL-L	BYOL+GI-CD-M	BYOL+G-GD-M	BYOL+GI-GD-M	BT-0.7-GL-M	BT-0.7+I-GL-M	BT-Rand-GL-M	BT-Rand+I-GL-M	BYOL+GI-GL-M	ATC-S	ATC+I-S	ST-DIM-S	ST-DIM+I-S	VAE-T-S	VAE-T+I-S
alien	·	·	+	−	·	−	−	·	−	+	+	·	·	−	+	·
amidar	·	·	·	+	·	+	·	·	·	·	+	+	+	·	·	+
assault	−	−	·	−	−	−	−	−	−	·	·	·	−	−	·	+
asterix	−	−	·	·	−	·	·	−	−	·	+	+	·	·	+	·
boxing	·	−	+	·	−	·	·	·	−	+	·	+	−	−	−	−
breakout	−	−	·	·	−	+	+	+	−	·	+	+	+	+	−	+
demon_attack	−	−	·	−	−	−	−	−	−	−	−	+	−	−	+	·
frostbite	·	·	·	+	·	·	·	+	·	·	−	·	·	·	+	+
gopher	+	−	·	·	+	+	+	·	·	·	+	+	+	+	+	·
krull	−	−	+	−	−	+	−	−	−	−	−	+	−	−	·	·
kung_fu_master	−	−	−	·	−	−	−	·	·	−	−	·	−	−	+	−
ms_pacman	−	−	·	−	+	·	·	·	·	+	·	−	·	·	·	+
pong	−	·	·	·	−	+	·	·	·	·	·	·	·	·	−	+
qbert	·	·	+	·	·	·	·	·	·	+	+	+	+	+	·	+
seaquest	−	−	−	−	−	−	−	−	−	·	·	+	·	−	·	−
Aggregate (IQM)	−	−	·	·	−	·	·	·	−	·	+	+	·	−	+	+
$-$ wins	9	10	2	6	9	5	6	5	8	3	4	1	5	7	3	3
$\cdot$ wins	5	5	9	7	4	5	7	8	7	8	5	5	6	5	6	5
$+$ wins	1	0	4	2	2	5	2	2	0	4	6	9	4	3	6	7

Table 27: Effect of the learning-rate sweep, in human-normalized score over the 240 (method, game) cells at the final pretraining checkpoint.

Per-cell difference	median	mean	p90	max
Best $-$ worst learning rate	0.090	0.272	0.901	2.505
Best $-$ default (10^{-4})	0.003	0.109	0.345	1.626

Project total. Combining the two tables gives roughly 5,700 GPU-h for pretraining, 36,900 for Rainbow finetuning, 11,800 for PPO finetuning, and 60 for probing: a project total of about 54,500 GPU-hours, or six GPU-years. Finetuning accounts for 89% of this and probing for 0.1%.

The ratio. Scoring a representation with the full proxy suite costs about 0.025 GPU-h, against 0.95 GPU-h for a single Rainbow seed on an S encoder and 2.80 on an L: between 1–3% of one seed, or a tenth of that against the ten-seed evaluation the paper actually uses to score a representation. Against PPO the fraction is smaller again, 0.2–0.7% of a seed. Pretraining is excluded from the comparison, as it is a sunk cost on both sides: the representation must exist before either can be applied.

G Sensitivity of the Probing Protocol

G.1 Probe Capacity: Linear versus MLP

In this ablation we study whether probe capacity changes conclusions on their ability to support ranking and model selection. We test this by re-running the experiment of cross-method selection with Rainbow and PPO (Sections 5.1 and 5.2) where now every learned proxy (with the exception

Table 28: Finetuning run counts.

Run set	Runs
Checkpoint sweep, Rainbow (10 ck × 10 methods × 15 games × 10 seeds)	15,000
Cross-method, Rainbow (16 methods × 15 games × 10 seeds × 3 LRs)	7,200
Shared between the two Rainbow sets (10 × 15 × 10)	−1,500
Cross-method, PPO (16 methods × 15 games × 10 seeds)	2,400
Total	23,100

Table 29: Median GPU-hours per run, by encoder size. Rainbow and PPO costs are per seed; every reported evaluation uses ten seeds. The probing figure covers the entire seven-proxy suite at one checkpoint, since feature extraction is shared across the proxies; it was timed in aggregate and so is not split by encoder size.

Stage	S	M	L
Pretraining (one method, game; 20 epochs)	3.9	33.2	46.2
Rainbow finetuning (one seed, 10^5 steps)	0.95	1.97	2.80
PPO finetuning (one seed, 10^7 steps)	3.51	4.5	10.8
Full proxy suite (one checkpoint)		0.025	

of effective rank) is a nonlinear model in place of the default linear model. The nonlinear model is the same architecture used for the Rainbow Q-head: Linear $\rightarrow$ ReLU $\rightarrow$ LayerNorm $\rightarrow$ Linear with hidden dimension 512.

We find that more capacity raises raw scores substantially on some probes but does not consistently improve their ability to rank methods, and never improves selection. Table 30 reports these quantities side by side.

The clearest cases are the terminal and reward probes, where the MLP raises the score most (0.23 to 1.00 and 0.55 to 0.93) yet top-1 selection degrades under both learners: from 0.918 to 0.715 (Rainbow) and 0.800 to 0.583 (PPO) for terminal, and from 1.000 to 0.918 and 1.000 to 0.800 for reward. Their rank correlations also mostly fall, with the exception of reward under PPO (+0.60 to +0.87). Elsewhere, rank correlation changes by between -0.25 and $+0.11$ with no consistent direction, and every other probe holds its NFoO_1 exactly under both learners.

Table 30: Probe capacity ablation. For each probe: the raw probe score (mean over the 15 games, averaged over methods), the Spearman ρ with downstream finetuning score, and NFoO_1. All at each method's final pretraining checkpoint over the full 16-method pool, using the same score/LR/aggregation rule as Section 5.1 – the *lin* column of every stat here is identical to the corresponding row of that section's tables.

	Probe score		ρ_{RB}		ρ_{PPO}		$\mathrm{NFoO}_1^{\mathrm{RB}}$		$\mathrm{NFoO}_1^{\mathrm{PPO}}$	
Probe	lin	MLP	lin	MLP	lin	MLP	lin	MLP	lin	MLP
Terminal	0.23	1.00	+0.89	+0.84	+0.61	+0.58	0.918	0.715	0.800	0.583
Reward	0.55	0.93	+0.87	+0.54	+0.60	+0.87	1.000	0.918	1.000	0.800
n-step return	0.55	0.54	+0.85	+0.86	+0.79	+0.82	0.918	0.918	0.800	0.800
Next action	0.24	0.25	+0.72	+0.81	+0.91	+0.73	0.918	0.918	0.800	0.800
Inverse action	0.34	0.36	+0.65	+0.76	+0.81	+0.79	0.715	0.715	0.583	0.583
Temporal distance	0.39	0.49	+0.66	+0.41	+0.82	+0.86	0.715	0.715	0.583	0.583

G.2 Probing data behavior policy

The proxies in the paper are trained on data that a DQN agent collected late in its own training, at its checkpoint 50 (except effective rank). Checkpoint 1 is data from the same agent while it was still weak. This section tests the seven proxies on the data of checkpoints 1, 10, 25 and 40 as well, on the final checkpoints of all 16 methods and 15 games, with 80k training and 20k validation transitions at

each. The encoder is always the method's final pretrained encoder; only the data changes. Table 31 asks whether a proxy trained on weak data ranks the 16 methods the way the same proxy trained on expert data does.

Table 31: **Seven proxies across probing-data checkpoints.** Each proxy is retrained on data from DQN checkpoint 1 (weak) up to 50 (near-expert), on the same 16 pretraining methods and 15 games, with the encoder fixed. Checkpoint 50 is the data used everywhere else in the paper. *Mean score* is the average over the 240 (method, game) cells; it shows how much the data changes what a proxy can predict (effective rank is a dimension count). *Agreement* is the Spearman ρ between the 16-method ranking a checkpoint produces and the one checkpoint 50 produces.

Data ckpt.	Terminal	Reward	n-step	Next act.	Inv. act.	Temp. dist.	Eff. rank
Mean score over the 240 *cells*							
1	0.238	0.515	0.421	0.277	0.388	0.389	2668
10	0.244	0.538	0.515	0.242	0.349	0.387	2696
25	0.215	0.546	0.523	0.242	0.347	0.379	2697
40	0.214	0.549	0.538	0.243	0.345	0.380	2702
50	0.228	0.552	0.547	0.243	0.343	0.386	2690
*Agreement with the checkpoint-*50 *ranking*							
1	0.959	0.982	0.956	0.988	0.988	1.000	1.000
10	0.982	0.991	0.994	1.000	0.997	0.994	0.999
25	0.962	0.991	0.988	0.991	0.997	0.997	0.997
40	0.968	0.997	0.985	0.994	0.991	0.985	0.997

Only n-step return changes much with the data, rising from 0.421 on the weakest to 0.547 on the reference; every other proxy stays within 0.045 of its checkpoint-50 score. The rankings agree throughout: the lowest agreement is 0.956, for n-step return on checkpoint 1; terminal stays between 0.959 and 0.982, and every other entry is at least 0.982. Which data a proxy is trained on is therefore not an important choice.

G.3 Probing dataset size

Every proxy in the main results is trained on 80k-transitions. Table 32 reruns the seven proxies with the training set cut to its first 20k, 40k and 60k transitions (nested budgets; the validation split is unchanged), on all 16 methods and 15 games. Effective rank is measured on the probing data here, like the other six proxies, so that the budget applies to it too; the main tables measure it on the pretraining data, which gives different scores but the same ordering of the 16 methods, so its 80k correlation coincides with Table 2. Scores are still rising at 80k for all six supervised proxies, so the budget is not past saturation, but what the proxies say about the encoders is fixed well before it: from 40k on, every proxy but terminal agrees with its 80k per-game rankings at 0.95 or better, and its correlation with finetuning moves by less than 0.05. Terminal is the exception because its positives are rare, tens to a couple of hundred per 80k transitions, so a small budget leaves the linear head few examples. The scores in the main text are therefore budget-limited in their absolute value but not in what they say about which representation to choose. The size of the probing dataset is therefore not a critical choice: from 40k transitions on, the proxies rank the encoders as they do at 80k.

G.4 The effect of n in n-step return

The n-step-return probe regresses onto the truncated discounted return $\sum_{k=0}^{n-1} \gamma^k r_{t+k}$ ($\gamma = 0.99$). The main text fixes $n = 10$; here we sweep $n \in \{1, 2, 3, 5, 10, 20, 40\}$ at each method's final checkpoint (Table 33). The mean R^2 over methods rises with n as the target accumulates reward variance, from 0.35 at $n = 1$ to a peak of 0.55 at $n = 10$ and $n = 20$, then eases to 0.52 at $n = 40$. Despite this, the quantities that matter for selection are stable. The method ranking at every n agrees with the $n = 10$ ranking at $\rho \geq 0.93$, and the probe-to-finetuning correlation stays within a narrow band for both learners ($\rho_{\text{RB}} \in [0.83, 0.92]$, $\rho_{\text{PPO}} \in [0.64, 0.79]$). The choice of $n = 10$ is therefore not load-bearing; any horizon from a few steps to a few tens gives the same cross-method verdict.

Table 32: **Sensitivity to the probe-training budget.** Seven proxies retrained on the first 20k/40k/60k transitions of each game, against the 80k runs used everywhere else; 16 pretraining methods, 15 games. *Mean score* is over the 240 (method, game) cells (effective rank is a dimension count, not a score in $[0, 1]$). *Agreement* is the within-game Spearman ρ between the 16-method ranking at that budget and at 80k, averaged over games.

	Mean score				Agreement w. 80k			ρ w. Rainbow finetune			
Proxy	20k	40k	60k	80k	20k	40k	60k	20k	40k	60k	80k
Terminal	0.140	0.191	0.214	0.228	0.619	0.743	0.823	+0.871	+0.879	+0.885	+0.888
Reward	0.508	0.536	0.547	0.552	0.916	0.952	0.968	+0.909	+0.915	+0.900	+0.874
n-step return	0.498	0.526	0.539	0.547	0.977	0.990	0.996	+0.897	+0.865	+0.853	+0.853
Next action	0.220	0.234	0.241	0.243	0.902	0.953	0.960	+0.738	+0.709	+0.726	+0.724
Inverse action	0.303	0.326	0.336	0.343	0.911	0.965	0.956	+0.624	+0.647	+0.647	+0.647
Temporal distance	0.307	0.348	0.371	0.386	0.917	0.963	0.977	+0.638	+0.650	+0.650	+0.662
Effective rank	2646	2683	2695	2690	0.991	0.994	0.993	+0.582	+0.582	+0.582	+0.582

Table 33: **The n-step-return probe as a function of the horizon n.** Linear head, each method's final checkpoint, 16 methods $\times$ 15 games. **Score**: mean over methods of the mean-over-games R^2. **Rank agreement**: Spearman ρ between the method ranking at n and at $n = 10$, the main-text value. **Predicts finetuning**: Spearman between the probe score and the finetuning score across the 16 methods.

		Rank agreement	Predicts finetuning	
n	Score $\overline{R^2}$	vs. n=10	Rainbow	PPO
1	0.35	0.93	+0.86	+0.70
2	0.43	0.97	+0.83	+0.79
3	0.47	0.99	+0.84	+0.79
5	0.51	0.99	+0.84	+0.79
10	0.55	1.00	+0.85	+0.79
20	0.55	0.97	+0.91	+0.69
40	0.52	0.94	+0.92	+0.64

G.5 Proxy variance

To analyze the variance of proxy tasks over random seeds, we re-fit each linear probe five times on the same frozen encoder, varying only the probe's random seed (Table 34). The mean within-cell standard deviation is at most 0.006 for every probe. Since a probe only needs to resolve differences between methods, the quantity that matters is seed noise relative to the between-method spread, i.e. the noise-to-signal ratio (*n:s*). It is at most 0.073, so re-fitting a probe with a different seed moves its score by a small fraction of the gap it is being used to detect. Recomputing the proxy-to-finetuning rank correlation separately for each seed confirms this: it varies by at most 0.05 under either learner.

H Sensitivity of Normalized Fraction-of-Oracle to the Candidate Pool

NFoO is computed relative to the best and worst methods in a candidate pool, so in principle its value depends on which methods the pool contains. If this dependence were strong, our reported values could be artifacts of our particular 16-method pool. We therefore measure how far NFoO moves under exhaustive perturbation of the pool.

We perform this evaluation on the Rainbow results of Section 5.1. For each proxy, we remove every j-method subset of the 16-method pool for $j \in \{1, \ldots, 10\}$, giving 58,650 subsets per proxy. For each subset, the proxy re-ranks the surviving methods, and NFoO_1 is recomputed against that subset's own extremes. We measure the deviation $\Delta = \mathrm{NFoO}_1(\text{subset}) - \mathrm{NFoO}_1(\text{full pool})$ and report two statistics: the *failure rate* (the percentage of subsets with $\Delta < -0.1$), and the mean of all

Table 34: **Proxy reliability across probe-training seeds.** We sweep the proxy tasks over seeds 1–5 for each representation. We report the mean within-cell standard deviation as *seed sd*; the median across games of the between-method standard deviation as *method sd*; their ratio as *n:s*. ρ_{RB} / ρ_{PPO} range is the min–max of the proxy to finetuning rank correlation recomputed per seed.

Probe	seed sd	method sd	n:s	ρ_{RB} rng	ρ_{PPO} rng
Terminal	0.0034	0.162	0.021	[+0.89, +0.91]	[+0.61, +0.61]
Reward	0.0007	0.146	0.005	[+0.87, +0.87]	[+0.60, +0.60]
n-step return	0.0001	0.117	0.001	[+0.85, +0.85]	[+0.79, +0.79]
Next action	0.0041	0.055	0.073	[+0.72, +0.74]	[+0.91, +0.92]
Inverse action	0.0041	0.084	0.049	[+0.65, +0.65]	[+0.81, +0.81]
Temporal distance	0.0055	0.105	0.053	[+0.63, +0.66]	[+0.80, +0.85]

negative deviations. We exclude positive deviations, which arise mechanically when the full-pool oracle is removed and the proxy's pick becomes the new ceiling.

Table 35: **Sensitivity of NFoO_1 to the candidate pool.** For each proxy: its value on the full 16-method pool; the percentage of subsets with $\Delta < -0.1$ after removing $j \in \{1, 5, 10\}$ methods; and the mean of all negative Δ, pooled over $j = 1, \ldots, 10$. Rainbow, each method's final pretraining checkpoint.

		% subsets with $\Delta < -0.1$			
Proxy	full pool	j=1	j=5	j=10	mean Δ when negative
Terminal	0.918	0%	0%	7%	−0.065
Reward	1.000	0%	2%	10%	−0.310
n-step return	0.918	0%	0%	2%	−0.029
Next action	0.918	6%	18%	20%	−0.151
Inverse action	0.715	0%	8%	24%	−0.178
Temporal distance	0.715	0%	2%	12%	−0.144
Effective rank	1.000	0%	8%	28%	−0.857

Small perturbations barely move the metric. Removing a single method causes a failure for only one proxy (next action, in 6% of subsets). With five of sixteen methods removed, the failure rate stays at or below 8% for every proxy except next action (18%). Failures rise only once most of the pool is discarded, reaching 2–28% at $j = 10$, where six methods remain. The values reported in the main text are therefore not an artifact of which candidates happen to be included.

Frequency and severity are separate axes. Reward fails in only 10% of subsets at $j = 10$, yet its negative deviations are the second largest (-0.310 on average). Effective rank is the extreme case: it fails only slightly more often than next action or inverse action (28% against 20% and 24% at $j = 10$), but its mean negative deviation is -0.857, roughly $2.8\times$ the next-worst proxy. The main text identifies the cause (Section 5.1): effective rank ranks the two ST-DIM variants 3rd and 4th, while their downstream performance places them 13th and 15th, and its full-pool NFoO_1 of 1.000 depends on neither being ranked first. Subsets that remove the methods above them promote an ST-DIM variant to the top, and that pick is close to the worst remaining method. Effective rank's failures are thus only slightly more frequent than other proxies', but far more severe.

I Early Stopping: Additional Results

This section reports additional results: Section I.1 sweeps the improvement threshold δ and Section I.2 reports post-hoc checkpoint selection and examines how much checkpoints vary within a run.

Table 36: **Early-stopping δ sweep.** Patience fixed at $p = 2$; $\star$ marks the default. Each cell is median compute% / median % of final-checkpoint score over the same 150 cells. The fixed-checkpoint row reports the closest-matching index, which varies by column (0.5%: idx 6, 1%: idx 5, 2%: idx 4, 4%: idx 4, 8%: idx 3). Random checkpoint selection reaches 93% of final-checkpoint score at 55% compute.

Proxy	δ=0.5%	δ=1%	δ=2%*	δ=4%	δ=8%
Terminal	50 / 97%	50 / 97%	50 / 97%	50 / 97%	50 / 97%
Reward	50 / 97%	50 / 96%	50 / 97%	40 / 97%	30 / 95%
n-step return	60 / 99%	50 / 96%	40 / 94%	30 / 92%	30 / 91%
Next action	50 / 95%	50 / 95%	40 / 94%	30 / 93%	30 / 91%
Inverse action	60 / 98%	50 / 97%	40 / 95%	30 / 94%	30 / 92%
Temporal distance	60 / 98%	50 / 95%	40 / 94%	30 / 94%	30 / 93%
Effective rank	60 / 100%	55 / 98%	50 / 98%	40 / 95%	30 / 94%
Validation loss	80 / 100%	70 / 100%	70 / 100%	70 / 100%	60 / 95%
Fixed checkpoint (compute-matched)	60 / 95%	50 / 94%	40 / 92%	40 / 92%	30 / 90%

I.1 Sensitivity to δ

Table 36 fixes patience p at 2 and sweeps the improvement threshold δ over $0.5\%, 1\%, 2\%, 4\%, 8\%$. We fix p rather than sweeping both parameters jointly because the two control the same underlying quantity: how much evidence of stagnation is required before halting.

Larger δ makes the rule more aggressive, since a larger improvement is needed to count as progress and the stagnation counter advances sooner. Compute falls from 50–60% at $\delta = 0.5\%$ to 30–50% at $\delta = 8\%$, and performance falls with it, from 95–100% to 91–97% of the final checkpoint's score. The terminal probe is the exception, unchanged across the entire range.

The comparison with the baselines is stable throughout. At every value of δ, all seven proxies match or exceed the closest-matching fixed checkpoint, so the main-text conclusion does not depend on the particular threshold chosen. At the default δ, every proxy also exceeds random checkpoint selection (93% at 55% compute) while using less compute. As in the main text, validation loss reaches higher performance (up to 100%) only by training longer, spending 60–80% of compute against 30–60% for the proxies.

I.2 Post-hoc checkpoint selection

Section 5.4 decides online when to stop a pretraining run. Here we study the post-hoc version of the problem: selecting among a run's ten checkpoints when all are available. Because this is a standard selection problem with a known oracle, we evaluate it with ρ and NFoO (Figure 7), rather than relative to the final checkpoint as in the online setting. We then examine how much a run's checkpoints actually differ, which explains both why ranking them is hard and why selecting among them is not.

For each of the 150 (method, game) cells in which every checkpoint was finetuned, we compute the *span*, the gap between the best and worst checkpoint on the seed-mean human-normalized score (HNS) curve. Figure 8 shows its distribution; we also measure how often the final checkpoint is the best and how much of the variance across checkpoints is attributable to seed noise.

Checkpoint variation is right-skewed. The median span is 0.153 HNS, and 26.7% of cells vary by less than 0.05 HNS across all ten checkpoints. The distribution has a long right tail (Figure 8), extending to a maximum span of 4.152 HNS and pulling the mean (0.373) to more than double the median. Still, 44.7% of cells have a span above 0.20 HNS, so meaningful variation is common. The final checkpoint is the best in only 26.0% of cells, which is why performance relative to the final checkpoint can exceed 100% in Section 5.4.

The best–worst gap is real, even though most checkpoint-to-checkpoint movement is noise. A run's span is typically several times larger than seed noise alone would produce (median $4.4\times$ the seed SEM), so it usually reflects a real difference. Yet seed noise accounts for a median of 60% of the

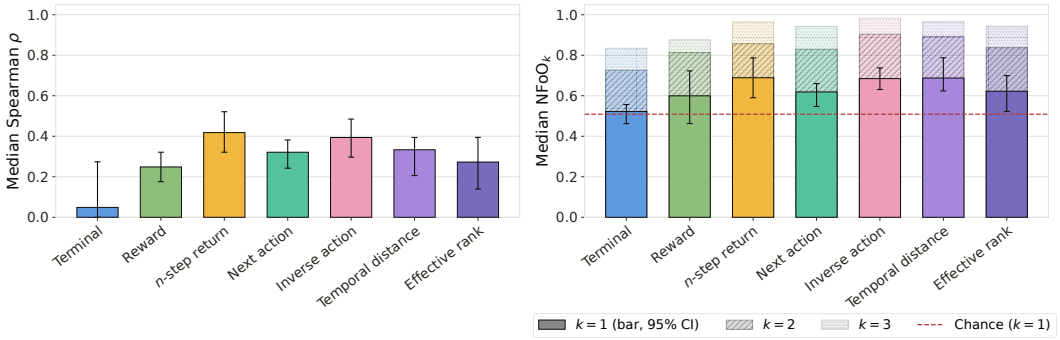

Figure 7: **Checkpoint-level ranking and selection.** Median Spearman ρ (left) and NFoO_k (right) over the 150 (method, game) cells, with the candidate pool restricted to the ten checkpoints of a single pretraining run. Error bars are 95% CIs for $k = 1$; hatched bars show $k = 2, 3$; the dashed line is chance at $k = 1$. Rainbow.

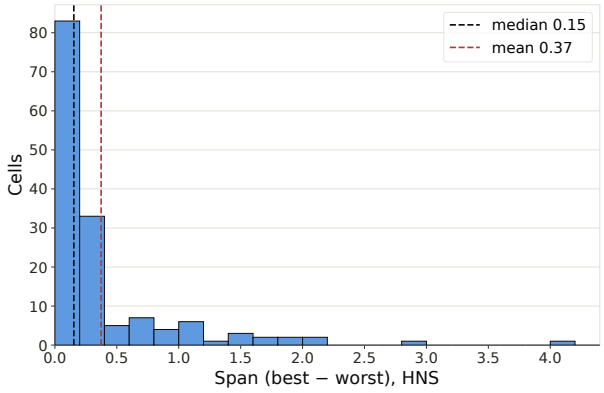

Figure 8: **Distribution of span (best − worst) across the 150 cells.** A single peak near zero with a smooth, monotonically decaying tail. The mean (0.37) is more than double the median (0.15).

variance across a run's checkpoints, and for more than half in 58% of cells. A typical run therefore has best and worst checkpoints that are clearly separated, while many intermediate checkpoints are largely interchangeable.

This explains the gap between the two panels of Figure 7. Spearman ρ penalizes every misordering equally, including those between checkpoints that differ by no more than seed noise, so no proxy exceeds $\rho = 0.41$. NFoO asks only whether the selected checkpoint is close to the best. Since most checkpoints cluster in the interchangeable middle, even random selection reaches $\mathrm{NFoO}_1 = 0.51$. Most proxies improve on this by 0.1–0.2, and with two or three picks reach NFoO above 0.8; terminal does not improve on chance.