# OPAIR: OBJECTIVE-GUIDED CAPACITY REPOPULATION FOR NEURON RECYCLING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Loss of plasticity limits the ability of deep reinforcement learning agents to keep learning, and neuron recycling mitigates it by periodically resetting under-utilized neurons. Existing methods focus on which neurons to release and reinitialize them with a rule independent of the training objective that zeroes their outgoing weights. The recycled neurons thus receive little learning signal from the objective and return to dormancy before they contribute to reducing it, which leaves the released capacity unused. To enable the released capacity to learn from the objective, we revisit neuron recycling as a two-stage procedure of release and repopulation and propose Objective-guided PAIRed neuron recycling (OPAIR). OPAIR adopts the activation-based release rule of ReDo, which we prove causes only a small objective perturbation, and replaces each pair of released neurons with two copies of an unreleased neuron, chosen by the objective and assigned opposite outgoing weights. This paired construction preserves the network function, and the objective-guided choice maximizes their objective-aligned learning signal along feature-rescaling directions among the candidate donors. Across 27 Atari and MuJoCo tasks, OPAIR achieves the highest mean final return on every task, and mechanism analyses confirm that both the paired construction and the objective-guided selection of copied neurons contribute.

## 1 INTRODUCTION

Loss of plasticity has been widely observed in deep reinforcement learning, where networks may gradually become less able to adapt over the course of training (Lyle et al., 2022; Nikishin et al., 2022; Kumar et al., 2020; Abbas et al., 2023). A growing line of work has addressed this problem through regenerative interventions, such as resetting network components, introducing new trainable capacity, or selectively replacing under-utilized features (Nikishin et al., 2022; Dohare et al., 2021; Nikishin et al., 2023; Farias & Jozefiak, 2024; Sokar et al., 2023). Among these approaches, neuron-level recycling offers a targeted way to recover under-utilized capacity while preserving the rest of the learned computation (Dohare et al., 2021; Farias & Jozefiak, 2024; Sokar et al., 2023).

Recent work on neuron-level recycling has focused on identifying under-utilized neurons and deciding when they should be recycled (Farias & Jozefiak, 2024; Sokar et al., 2023; Liu et al., 2026), using activation statistics in ReDo (Sokar et al., 2023) or gradient magnitude in ReGraMa (Liu et al., 2026). Both reinitialize the released neurons with a rule independent of the training objective, which sets their outgoing weights to zero. Their incoming parameters receive no gradient until the outgoing weights move away from zero, and the recycled neurons are repeatedly reset before they learn much from the objective. Since recycling aims to restore the ability of the network to fit the changing targets of its training objective (Lyle et al., 2022; Abbas et al., 2023; Dohare et al., 2024), capacity that is reset before learning from this objective is left largely unused.

To enable the released capacity to learn from the objective, we revisit neuron recycling as a two-stage procedure of release and repopulation and assess both stages by the training objective. We prove that the activation-based release of ReDo causes only a small objective perturbation, so the effect of recycling is determined by repopulation, which we formalize as preserving the network function while maximizing the first-order objective reduction available to the recycled neurons along feature-rescaling directions. We introduce **OPAIR**, an **O**bjective-guided **PAIR**ed neuron recycling method. A single recycled neuron cannot preserve the network function and receive gradient at the



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.

same time, since both its forward contribution and the gradient of its incoming parameters scale with its outgoing weights. OPAIR therefore replaces each pair of released neurons with two copies of an unreleased neuron with opposite outgoing weights, which resolves this conflict, and chooses the copied neurons to maximize this criterion. Across Atari-17 with DQN and MuJoCo-10 with SAC and the BRO-Net architecture, OPAIR achieves the highest mean final return on all 27 tasks.

Our contributions are threefold:

- We revisit neuron recycling as a two-stage procedure assessed by the training objective and show that the effect of recycling is determined by repopulation.
- We propose OPAIR, which repopulates released neurons with objective-selected paired copies that preserve the network function and maximize their first-order objective reduction along feature-rescaling directions among the candidate donors.
- Across 27 Atari and MuJoCo tasks, OPAIR achieves the highest mean final return, and mechanism analyses confirm the contribution of each component.

## 2 Revisiting Neuron Recycling: Release and Repopulation

Neuron recycling periodically replaces under-utilized neurons so that their capacity can participate in subsequent learning. Each recycling event involves two decisions, namely which neurons to release and how to repopulate the released slots. The left side of Figure 1 illustrates this process with ReDo as an example. Existing methods have progressively refined the release signal, from activation statistics in ReDo and Self-Normalized Resets (Sokar et al., 2023; Farias & Jozefiak, 2024) and the contribution utility of Continual Backprop (Dohare et al., 2021; 2024) to gradient magnitude in ReGraMa (Liu et al., 2026). Most of these methods reinitialize the incoming parameters of released neurons and set their outgoing weights to zero, and ReBorn transfers weights from over-active neurons while preserving the relative ordering of action values (Qin et al., 2024). This line of work has thus concentrated on the release decision and treated repopulation as a fixed rule. We revisit neuron recycling as a two-stage procedure of release and repopulation.

### 2.1 Neuron Recycling and the Training Objective

Consider an $L$-hidden-layer feed-forward network with $a^0 = x$, $z^\ell = W^\ell a^{\ell-1} + b^\ell$, and $a^\ell = \sigma(z^\ell)$ for $\ell = 1, \ldots, L$, and output $f_\theta(x) = W^{L+1}a^L + b^{L+1}$. The optimization objective is $\mathcal{J}(F) = \mathbb{E}_{x\sim\mathcal{D}}[\mathcal{L}(F(x))]$, where $\mathcal{D}$ denotes the current training distribution. In deep reinforcement learning, $\mathcal{J}$ is the training loss, such as the temporal-difference loss of DQN or the critic loss of SAC.

Let $F_{\text{old}}$ denote the network immediately before a recycling event. Releasing a set of neuron slots removes the learned functions currently occupying those slots and produces an intermediate network $F_{\text{mask}}$. Repopulating the released slots then produces the post-recycling network $F_{\text{rec}}$:

$$F_{\text{old}} \xrightarrow{\text{release}} F_{\text{mask}} \xrightarrow{\text{repopulation}} F_{\text{rec}}. \tag{1}$$

Prior work characterizes loss of plasticity in deep reinforcement learning as a declining ability to fit the changing targets of $\mathcal{J}$ (Lyle et al., 2022; Kumar et al., 2020; Abbas et al., 2023; Dohare et al., 2024). Neuron recycling restores this ability by replacing part of the learned computation, which risks discarding what the network has learned, as resets that discard learned computation cause performance drops (Nikishin et al., 2022; Kim et al., 2023; Lee et al., 2024). We therefore use $\mathcal{J}$ itself to assess both steps in Equation 1. Release should perturb $\mathcal{J}$ as little as possible, and repopulation should give the released capacity as strong a learning signal from $\mathcal{J}$ as possible in subsequent optimization.

### 2.2 Stage I: Safe Capacity Release

Existing release signals are neuron-level statistics whose relation to $\mathcal{J}$ is left implicit. We make this relation explicit by quantifying release through the change it causes in $\mathcal{J}$. For a set $S_\ell$ of neuron slots considered for release in layer $\ell$, we define the *release perturbation* as

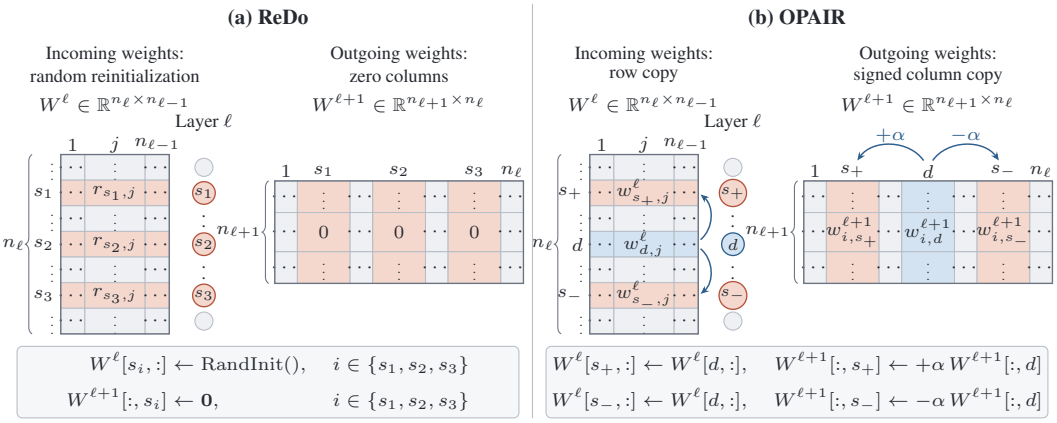



Figure 1: Parameter-level comparison of ReDo and OPAIR in layer $\ell$. The left side illustrates ReDo, which reinitializes the parameters of released neuron slots. The right side illustrates OPAIR zero-pair repopulation, which copies the donor row into both released slots and initializes their outgoing columns as $+\alpha$ and $-\alpha$ copies of the donor column.

$$\left|\mathcal{J}(F^{S_\ell}_{\text{mask}}) - \mathcal{J}(F_{\text{old}})\right|, \tag{2}$$

where $F^{S_\ell}_{\text{mask}}$ denotes the network obtained by removing the current functions associated with $S_\ell$. Stage I seeks sets for which this perturbation is small.

For neuron $s$ in layer $\ell$, let $h^\ell_s(x)$ denote its activation and let $v^\ell_s = W^{\ell+1}[:,s]$ denote its outgoing vector. Its direct contribution to the next-layer pre-activation is therefore $h^\ell_s(x)v^\ell_s$.

**Proposition 1** (Low activation bounds the release perturbation)**.** *Let $F^{(s)}_{\text{mask}}$ denote the network obtained by removing neuron $s$ from layer $\ell$. Suppose that $\|v^\ell_s\| \le B_\ell$, the downstream mapping from $z^{\ell+1}$ to the network output is locally $\lambda_\ell$-Lipschitz, and the loss is locally $\kappa$-Lipschitz with respect to the network output. Then*

$$\left|\mathcal{J}(F^{(s)}_{\text{mask}}) - \mathcal{J}(F_{\text{old}})\right| \le \kappa\lambda_\ell B_\ell \mathbb{E}_{x\sim\mathcal{D}}\left[|h^\ell_s(x)|\right]. \tag{3}$$

The proof is given in Appendix A.1, and Appendix A.2 extends the result to sets of neurons and multiple layers. ReDo (Sokar et al., 2023) motivates its release rule by the intuition that low-activation neurons have little effect on the network output. Proposition 1 proves that low average activation with bounded outgoing weights guarantees a small release perturbation, and thus provides an objective-level justification for this rule.

### 2.3 Stage II: Capacity Repopulation

Repopulation determines the *learning signal* that the released capacity receives in subsequent optimization. This signal acts through gradient updates of the repopulated slots, so we examine the gradient that reaches them. Consider a released slot $s$ in layer $\ell$ with incoming weights $W^\ell[s,:]$, and let $\delta^{\ell+1}(x) = \partial\mathcal{L}/\partial z^{\ell+1}(x)$ denote the gradient at the next-layer pre-activation. The gradient of the incoming weights factorizes as

$$\nabla_{W^\ell[s,:]}\mathcal{L} = \underbrace{\left((v^\ell_s)^\top \delta^{\ell+1}(x)\right)}_{\text{gradient through } v^\ell_s} \underbrace{\sigma'\left(z^\ell_s(x)\right)}_{\text{activity of } s} a^{\ell-1}(x)^\top. \tag{4}$$

ReDo and ReGraMa (Sokar et al., 2023; Liu et al., 2026) repopulate a released slot by reinitializing its incoming parameters and setting its outgoing vector to zero. Under this rule, $v^\ell_s = 0$ makes the first factor vanish until the outgoing vector grows away from zero, and the incoming parameters, drawn independently of $\mathcal{J}$, leave the second factor to chance. The repopulated slot therefore receives little learning signal and tends to return to the low-activity state that triggered its release. Consistent

with this analysis, ReDo neurons return to the low-activity state and are released again after only 2–6 recycling events on average (Section 4.3), so the same slots are repeatedly released and repopulated.

**Repopulation Formalization.** We now formalize Stage II. A repopulation $q$ assigns new incoming parameters $\psi_s$, including biases, and new outgoing vectors $v_s^\ell$ to the released slots $s \in S$ while leaving all other parameters unchanged, producing the network $F_{\mathrm{rec}}(\cdot\,; q)$. Equation 4 shows that the learning signal of a repopulated slot depends jointly on its activity and on the objective gradient propagated through its outgoing vector. To quantify this signal at the level of the training objective, for each released slot $s$ we introduce a rescaling coordinate $\beta_s$ that replaces its feature $h_s^\ell(x)$ by $(1+\beta_s)h_s^\ell(x)$. Let

$$\widetilde{\mathcal{J}}_q(\boldsymbol{\beta}) = \mathbb{E}_{x\sim\mathcal{D}}\left[\widetilde{\mathcal{L}}_x(\boldsymbol{\beta}; q)\right] \tag{5}$$

denote the resulting training objective. At $\boldsymbol{\beta} = 0$,

$$\left.\frac{\partial \widetilde{\mathcal{J}}_q}{\partial \beta_s}\right|_{\boldsymbol{\beta}=0} = \mathbb{E}_{x\sim\mathcal{D}}\left[h_s^\ell(x)\frac{\partial \mathcal{L}(F_{\mathrm{rec}}(x;q))}{\partial h_s^\ell(x)}\right]. \tag{6}$$

For positively one-homogeneous activations such as ReLU, this derivative is also $\psi_s^\top \nabla_{\psi_s}\mathcal{J}$, the directional derivative of the training objective along a rescaling of the incoming parameters (Appendix A.4). Consider a gradient step of size $\eta$ along these rescaling coordinates, $\boldsymbol{\beta}' = -\eta\nabla_{\boldsymbol{\beta}}\widetilde{\mathcal{J}}_q(0)$. Its *first-order objective reduction* is

$$\Delta(q) = \eta\sum_{s\in S}\left(\mathbb{E}_{x\sim\mathcal{D}}\left[h_s^\ell(x)\frac{\partial \mathcal{L}(F_{\mathrm{rec}}(x;q))}{\partial h_s^\ell(x)}\right]\right)^2 \tag{7}$$

Indeed, $\widetilde{\mathcal{J}}_q(0) - \widetilde{\mathcal{J}}_q(\boldsymbol{\beta}') = \Delta(q) + O(\eta^2)$. Thus, $\Delta(q)$ measures the first-order decrease of the current training objective available to the released slots along feature-rescaling directions. Since $\Delta(q)$ grows quadratically with the scales of the new incoming parameters and outgoing vectors, we compare repopulations at bounded scales, under which the maximization favors better features over larger parameters and the first-order approximation behind $\Delta(q)$ remains accurate (Appendix A.5). Let $\mathcal{Q}$ denote the set of repopulations whose incoming parameters satisfy $\|\psi_s\| \le C_\ell$ and whose outgoing vectors take the form $v_s^\ell = \alpha v$ with $\|v\| \le B_\ell$, where $\alpha > 0$ is a scale hyperparameter, and $C_\ell$ and $B_\ell$ bound the incoming and outgoing norms of the neurons in layer $\ell$, with $B_\ell$ as in Proposition 1. Stage II seeks the repopulation with the largest first-order objective reduction among those that leave the network established after release unchanged:

$$q^\star \in \arg\max_{q\in\mathcal{Q}} \Delta(q) \quad \text{subject to} \quad R_{\mathrm{mask}}(q) = 0, \tag{8}$$

where

$$R_{\mathrm{mask}}(q) = \mathbb{E}_{x\sim\mathcal{D}}\left[\|F_{\mathrm{rec}}(x;q) - F_{\mathrm{mask}}(x)\|_2^2\right]. \tag{9}$$

By Appendix A.3, $R_{\mathrm{mask}}$ bounds the repopulation perturbation $|\mathcal{J}(F_{\mathrm{rec}}) - \mathcal{J}(F_{\mathrm{mask}})|$, so the constraint leaves $\mathcal{J}$ unchanged by repopulation, and a recycling event changes $\mathcal{J}$ only through the release perturbation of Stage I. The rule of ReDo and ReGraMa sets $v_s^\ell = 0$, which satisfies the constraint but gives $\partial\mathcal{L}/\partial h_s^\ell = (v_s^\ell)^\top\delta^{\ell+1} = 0$ and hence $\Delta(q) = 0$. These methods thus satisfy the constraint in Equation 8 without addressing its maximization.

## 3 OPAIR: Objective-Guided Paired Neuron Recycling

Section 2 formalizes the two stages of neuron recycling, release and repopulation. For Stage I, we prove in Proposition 1 that low activation guarantees a small release perturbation, an objective-level justification for activation-based release. OPAIR therefore follows ReDo (Sokar et al., 2023) and releases low-activity neurons. In this section, we solve the repopulation problem of Stage II in Equation 8 in two steps. Section 3.1 constructs the form of the solution, zero-pair repopulation, which satisfies the constraint while allowing a positive first-order objective reduction. Section 3.2 then solves the maximization within this form, which yields objective-guided pair selection. Section 3.3 presents the complete OPAIR algorithm.

**Algorithm 1** OPAIR: Objective-Guided Paired Neuron Recycling

**Require:** Network parameters $\theta$, optimizer state $m$, replay buffer $\mathcal{B}$, recyclable layers $\mathcal{R}$
**Require:** Activity threshold $\tau$, zero-pair scale $\alpha$
**Ensure:** Updated parameters and optimizer state $(\theta, m)$
Sample a scoring minibatch $\mathcal{X}$ uniformly from $\mathcal{B}$
**for** each recyclable layer $\ell \in \mathcal{R}$ **do**
  ▷ *Stage I: Safe capacity release*
  Compute normalized activity scores $\{r_i^\ell\}$
  $S_\ell \leftarrow$ the $2N_\ell$ lowest-activity neurons with $r_i^\ell < \tau$
  ▷ *Stage II: Objective-guided capacity repopulation*
  Compute $u_d^\ell$ for all $d \in A_\ell$ (Eq. 11)
  $D_\ell \leftarrow \mathrm{TopN}_{d \in A_\ell}\ u_d^\ell$ (Eq. 13)
  **for** each donor $d_j \in D_\ell$ **do**
    Assign two released slots $(s_j^+, s_j^-) \in S_\ell$
    $\psi_{s_j^+}, \psi_{s_j^-} \leftarrow \psi_{d_j}$
    $v_{s_j^+}^\ell \leftarrow +\alpha v_{d_j}^\ell,\ v_{s_j^-}^\ell \leftarrow -\alpha v_{d_j}^\ell$
  **end for**
**end for**
Reset optimizer states associated with all modified parameters
**return** $(\theta, m)$

### 3.1 Zero-Pair Repopulation

Section 2.3 shows that the rule of ReDo and ReGraMa (Sokar et al., 2023; Liu et al., 2026) satisfies the constraint in Equation 8 with $\Delta(q) = 0$. More generally, a repopulation that treats each released slot individually cannot satisfy the constraint in Equation 8 with a positive $\Delta(q)$, since the constraint then requires each slot to be inactive or to have a zero outgoing vector, and by Equation 4 either condition removes its learning signal. A positive $\Delta(q)$ under the constraint therefore requires several released slots whose nonzero contributions cancel one another, so that the constraint holds while each slot retains a nonzero learning signal. We therefore propose *zero-pair repopulation*.

In zero-pair repopulation, we construct two released slots as a pair $(s_+, s_-)$, which share the same incoming parameters $\psi$ and receive opposite outgoing vectors $\pm\alpha v$:

$$\psi_{s_+} = \psi_{s_-} = \psi, \qquad v_{s_+}^\ell = +\alpha v, \qquad v_{s_-}^\ell = -\alpha v. \tag{10}$$

The two slots compute the same feature and have opposite outgoing vectors, so their contributions to the next layer cancel and the network after repopulation coincides with $F_{\mathrm{mask}}$, which satisfies the constraint in Equation 8. The cancellation holds for the pair as a whole, while each slot keeps a nonzero outgoing vector. By Equation 4, each slot can therefore receive a nonzero learning signal. Its contribution to $\Delta(q)$ is positive whenever the expected activation–gradient product is nonzero.

Appendix A.6 extends this function preservation to simultaneous recycling in multiple layers. Zero-pair repopulation thus satisfies the constraint in Equation 8 for every choice of $(\psi, v)$, and Equation 8 reduces to choosing these parameters to maximize $\Delta(q)$.

### 3.2 Objective-Guided Pair Selection

**Closed form of the first-order objective reduction.** We solve this maximization by writing $\Delta(q)$ in closed form under zero-pair repopulation. Let the $2N$ released slots of layer $\ell$ form $N$ pairs with parameters $(\psi_j, v_j)$, $j = 1, \ldots, N$, let $h_\psi(x)$ denote the feature computed by a pair, and let $g_v(x) = v^\top \delta^{\ell+1}(x)$ denote the objective gradient propagated through $v$. For a zero-pair with outgoing vectors $+\alpha v$ and $-\alpha v$, the two rescaling-coordinate gradients of the training objective are $+\alpha \mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x) g_v(x)]$ and $-\alpha \mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x) g_v(x)]$. Their squared magnitudes sum to $2\alpha^2 (\mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x) g_v(x)])^2$, so the first-order objective reduction becomes

$$\Delta(q) = 2\eta\alpha^2 \sum_{j=1}^{N} u(\psi_j, v_j), \qquad u(\psi, v) = \left(\mathbb{E}_{x\sim\mathcal{D}}\left[h_\psi(x) g_v(x)\right]\right)^2. \tag{11}$$

Appendix A.6 gives the derivation. The step size $\eta$ and the scale $\alpha$ are fixed, so maximizing $\Delta(q)$ is equivalent to maximizing $\sum_j u(\psi_j, v_j)$. We call $u(\psi, v)$ the *objective sensitivity* of a pair.

**From continuous search to neuron selection.** Under the scale bounds of $\mathcal{Q}$, maximizing Equation 11 is a nonconvex problem over $(\psi, v)$. We propose to solve it by selecting among the unreleased neurons in layer $\ell$, denoted by $A_\ell$, for three reasons. Their incoming and outgoing norms lie within the bounds $C_\ell$ and $B_\ell$ of $\mathcal{Q}$ (Section 2.3), so copying them keeps the pairs in $\mathcal{Q}$. A single forward and backward pass on a minibatch yields the activation $h_d^\ell$ and the activation gradient $g_d^\ell$ of every unreleased neuron $d$, so the objective sensitivity of every candidate can be estimated. Training has optimized their parameters under $\mathcal{J}$, and release retains them because they compute active features, so the pairs start from features that the objective relies on. The unreleased neurons form a finite candidate set within $\mathcal{Q}$ over which the estimated criterion can be maximized exactly.

We restrict the maximization to pairs that copy distinct unreleased neurons, with $\psi = \psi_d$ and $v = v_d^\ell = W^{\ell+1}[:, d]$ for $d \in A_\ell$, which we call *donors*. Since $g_d^\ell(x) = \partial \mathcal{L} / \partial h_d^\ell(x)$, each donor has objective sensitivity

$$u_d^\ell = u(\psi_d, v_d^\ell) = \left(\mathbb{E}_{x \sim \mathcal{D}} \left[h_d^\ell(x) g_d^\ell(x)\right]\right)^2. \tag{12}$$

The maximization becomes a discrete choice of $N$ donors. Since the donors contribute additively to $\Delta(q)$ in Equation 11, the maximization is solved by the $N$ donors with the largest objective sensitivity,

$$D_\ell = \mathrm{TopN}_{d \in A_\ell} u_d^\ell \in \arg \max_{\substack{D \subseteq A_\ell \\ |D| = N}} \sum_{d \in D} u_d^\ell. \tag{13}$$

We call this rule *objective-guided pair selection*, in which each zero-pair copies one of the $N$ unreleased neurons with the largest objective sensitivity, as illustrated on the right side of Figure 1. Section 3.3 turns this solution into the complete OPAIR algorithm.

Table 1: Atari-17 final evaluation return at 2.5M environment transitions. Entries are mean ± standard deviation across five seeds; bold marks the largest unrounded mean in each row.

| Task | Base | ReDo | ReGraMa | OPAIR |
|---|---|---|---|---|
| Asterix | 1403.9 ± 179.8 | 2600.6 ± 49.5 | 1455.7 ± 100.1 | **3237.1 ± 180.3** |
| BeamRider | 2501.6 ± 92.0 | 5383.6 ± 222.4 | 2363.8 ± 41.6 | **7222.7 ± 150.8** |
| Breakout | 48.6 ± 0.8 | 53.8 ± 2.8 | 60.9 ± 4.5 | **73.8 ± 3.5** |
| DemonAttack | 584.5 ± 30.2 | 2940.2 ± 777.8 | 934.5 ± 166.9 | **7613.3 ± 603.8** |
| DoubleDunk | −18.2 ± 0.5 | −16.2 ± 1.2 | −16.0 ± 0.4 | **−12.9 ± 1.1** |
| Enduro | 690.2 ± 61.2 | 548.3 ± 167.9 | 690.4 ± 108.6 | **961.4 ± 86.3** |
| IceHockey | −13.0 ± 0.8 | −6.1 ± 0.5 | −8.6 ± 0.6 | **−5.1 ± 0.3** |
| Jamesbond | 192.1 ± 29.8 | 382.1 ± 54.2 | 325.5 ± 16.8 | **500.4 ± 14.2** |
| MsPacman | 1753.9 ± 92.5 | 2206.6 ± 113.1 | 1878.2 ± 79.5 | **2705.2 ± 159.7** |
| Pong | 17.0 ± 0.7 | 11.5 ± 2.0 | 15.8 ± 0.5 | **18.9 ± 0.6** |
| Qbert | 4283.3 ± 231.1 | 3746.7 ± 140.7 | 4096.9 ± 426.6 | **5066.2 ± 103.7** |
| RoadRunner | 37806.2 ± 267.7 | 43138.8 ± 637.3 | 39953.4 ± 1623.8 | **44461.0 ± 721.9** |
| Seaquest | 1711.1 ± 114.5 | 2252.7 ± 467.3 | 1606.6 ± 236.5 | **3788.9 ± 243.4** |
| SpaceInvaders | 530.7 ± 26.8 | 647.1 ± 31.0 | 577.0 ± 22.2 | **740.7 ± 15.2** |
| WizardOfWor | 907.5 ± 490.2 | 679.4 ± 668.0 | 1582.4 ± 134.3 | **2820.7 ± 119.3** |
| YarsRevenge | 7979.6 ± 612.4 | 10149.7 ± 2186.4 | 9572.0 ± 2130.2 | **15658.4 ± 1208.0** |
| Zaxxon | 1747.6 ± 2189.9 | 2576.8 ± 1323.7 | 5013.4 ± 278.4 | **5436.8 ± 195.3** |

### 3.3 The Complete OPAIR Algorithm

OPAIR is inserted periodically into an otherwise unchanged RL optimization loop, and Algorithm 1 summarizes one recycling event. Line 1 samples a minibatch from the replay buffer, on which activity scores and objective sensitivities are estimated under the current RL objective. For each donor, $u_d^\ell$ is estimated by squaring the minibatch mean of $h_d^\ell g_d^\ell$, as specified in Appendix B.1. Lines 3–4 follow ReDo and treat neurons whose normalized activity score $r_s^\ell$ falls below a threshold $\tau$ as recyclable candidates. Each zero-pair occupies two released slots, so with $K_\ell$ candidates in layer $\ell$, OPAIR sets $N_\ell = \lfloor K_\ell / 2 \rfloor$ and releases the $2N_\ell$ lowest-activity candidates $S_\ell = \{s_1, \ldots, s_{2N_\ell}\}$. Lines 5–6 select $N_\ell$ donors by Equation 13, and Lines 8–10 build one zero-pair per donor as in Equation 10. The scale $\alpha$ controls the magnitude of the learning signals while preserving pairwise

cancellation, and it leaves the ranking of donors unchanged. The two copies are subsequently optimized as independent parameters, and Line 13 clears the optimizer history associated with the replaced neuron functions.

# 4 EXPERIMENTS

Our experiments address four questions. (1) Does OPAIR improve performance across tasks, algorithms, and architectures? (2) Do neurons recycled by OPAIR re-enter subsequent optimization? (3) How do objective-guided donor selection and zero-pair repopulation contribute to its performance? (4) How sensitive is OPAIR to the hyperparameters governing release and repopulation?

## 4.1 EXPERIMENTAL SETUP

We study visual discrete control on Atari-17 with DQN and state-based continuous control on MuJoCo-10 with SAC and BRO-Net critics, so the two benchmarks differ in observation space, action space, and network architecture. Atari agents use a convolutional Q-network and train for $2.5$ million environment transitions ($10$ million emulator frames), and MuJoCo agents train for $1$ million environment transitions.

We compare Base, which trains the same agent without recycling; activation-based ReDo (Sokar et al., 2023); gradient-based ReGraMa (Liu et al., 2026); and OPAIR. Within each benchmark, methods share the architecture, optimization settings, interaction budget, and evaluation schedule. We report final evaluation return as mean $\pm$ standard deviation across five seeds. Appendix B specifies the environments, training and recycling configurations, and evaluation protocol.

## 4.2 OVERALL PERFORMANCE

OPAIR achieves the highest mean on all $17$ Atari games and all $10$ MuJoCo tasks (Tables 1 and 2). On Atari, the margins are largest on games with more room for improvement, where OPAIR improves over the strongest comparator by $24\%$ on Asterix, $78\%$ on WizardOfWor, and $159\%$ on DemonAttack. On MuJoCo, all methods are near saturation on InvertedDoublePendulum, while OPAIR shows more evident advantages on HumanoidStandup, Hopper, Swimmer, and Walker2d. Learning curves for all tasks are provided in Appendices B.2 and B.3.

Table 2: MuJoCo-10 final evaluation return at 1M environment transitions. The statistic and boldface convention match Table 1.

| Task | Base | ReDo | ReGraMa | OPAIR |
|---|---|---|---|---|
| Ant | $6723.3 \pm 139.2$ | $5441.7 \pm 231.1$ | $3293.3 \pm 567.7$ | $\mathbf{7064.4 \pm 42.4}$ |
| HalfCheetah | $11596.7 \pm 339.8$ | $10863.6 \pm 122.1$ | $9326.7 \pm 703.3$ | $\mathbf{12708.5 \pm 321.4}$ |
| Hopper | $1820.8 \pm 361.4$ | $3147.0 \pm 54.4$ | $3067.3 \pm 244.7$ | $\mathbf{3683.0 \pm 238.4}$ |
| Humanoid | $5162.3 \pm 62.5$ | $5199.0 \pm 67.5$ | $5080.3 \pm 234.0$ | $\mathbf{5380.7 \pm 32.1}$ |
| HumanoidStandup | $150906.4 \pm 4175.1$ | $155594.0 \pm 3523.9$ | $153059.0 \pm 1927.9$ | $\mathbf{181408.9 \pm 3339.8}$ |
| InvertedDoublePendulum | $9357.4 \pm 0.8$ | $9355.3 \pm 1.9$ | $9354.2 \pm 2.3$ | $\mathbf{9359.1 \pm 2.2}$ |
| Pusher | $-22.0 \pm 0.3$ | $-22.6 \pm 0.2$ | $-23.4 \pm 0.7$ | $\mathbf{-20.7 \pm 0.3}$ |
| Reacher | $-3.8 \pm 0.2$ | $-3.8 \pm 0.2$ | $-4.1 \pm 0.3$ | $\mathbf{-3.1 \pm 0.2}$ |
| Swimmer | $111.9 \pm 13.0$ | $52.8 \pm 33.2$ | $49.5 \pm 36.4$ | $\mathbf{132.4 \pm 3.8}$ |
| Walker2d | $4659.2 \pm 112.9$ | $4213.0 \pm 194.8$ | $4363.5 \pm 180.7$ | $\mathbf{5238.7 \pm 183.8}$ |

## 4.3 RE-ENTRY OF RECYCLED CAPACITY

We track the *re-reset interval* of every recycled neuron, the number of recycling events between two consecutive releases, which measures how quickly recycled capacity re-enters subsequent optimization (Table 5 in Appendix B.4). On Asterix, Seaquest, Ant, and Walker2d, ReDo neurons are re-reset after only 2–6 recycling events on average, and ReDo accumulates more than $100$ times as many resets over training as OPAIR, consistent with the analysis in Section 2.3. OPAIR-Zero, which copies the same donor's incoming weights and biases but initializes both outgoing vectors to zero, isolates the effect of donor-based initialization and reaches intervals of $101$–$309$ recycling events. OPAIR assigns oppositely signed outgoing vectors and extends the interval to $149$–$508$ recycling events on

all four tasks, while OPAIR-Zero performs 1.8–3.4 times as many resets as OPAIR. The nonzero learning signal therefore keeps recycled neurons active longer than donor-based initialization alone.

### 4.4 DISSECTING OPAIR

**Donor selection.** To assess the role of objective-guided donor selection, which maximizes the first-order objective-reduction criterion in Equation 8, we compare OPAIR with three alternative donor-selection strategies while keeping the capacity-release rule and zero-pair repopulation unchanged. Random Donor samples donors uniformly from the remaining neurons $A_\ell$, Activation Donor ranks them by absolute activation, and Gradient Donor ranks them by absolute activation-gradient magnitude. OPAIR ranks donors by objective sensitivity (Equation 12). In Table 3, OPAIR achieves the highest mean on all seven tasks. Among the donor variants, the strongest changes from task to task, with each of Random, Activation, and Gradient Donor ranking first among them on at least two tasks. Activation Donor and Gradient Donor each use only one factor, whereas OPAIR measures their signed alignment through the expected activation–gradient product.

Table 3: Mechanism analysis: final evaluation return, mean ± standard deviation across five seeds, rounded to integers. Bold marks the largest mean in each column.

| | Atari | | | | MuJoCo | | |
|---|---|---|---|---|---|---|---|
| Method | Asterix | BeamRider | DemonAttack | Seaquest | Ant | Walker2d | Humanoid |
| *Donor selection* | | | | | | | |
| Random | $1899 \pm 297$ | $5936 \pm 212$ | $2629 \pm 813$ | $2641 \pm 213$ | $6440 \pm 334$ | $2610 \pm 1301$ | $4996 \pm 142$ |
| Activation | $2427 \pm 51$ | $4786 \pm 432$ | $4218 \pm 572$ | $2391 \pm 406$ | $4926 \pm 1862$ | $3800 \pm 640$ | $5231 \pm 95$ |
| Gradient | $2088 \pm 150$ | $5150 \pm 754$ | $1239 \pm 175$ | $2235 \pm 334$ | $6676 \pm 206$ | $4302 \pm 297$ | $4390 \pm 427$ |
| *Paired repopulation* | | | | | | | |
| OPAIR-Zero | $1918 \pm 212$ | $3874 \pm 1403$ | $2080 \pm 521$ | $2171 \pm 27$ | $5128 \pm 1046$ | $3460 \pm 832$ | $5182 \pm 124$ |
| OPAIR-Same | $2544 \pm 353$ | $5728 \pm 355$ | $3656 \pm 1048$ | $2446 \pm 314$ | $6506 \pm 398$ | $4393 \pm 154$ | $5194 \pm 263$ |
| *Reset budget* | | | | | | | |
| ReDo (2% cap) | $1953 \pm 128$ | $5134 \pm 1148$ | $1966 \pm 641$ | $1938 \pm 1187$ | $6860 \pm 1732$ | $4813 \pm 429$ | $5193 \pm 183$ |
| ReGraMa (2% cap) | $1410 \pm 226$ | $2568 \pm 159$ | $743 \pm 194$ | $1243 \pm 605$ | $2878 \pm 1236$ | $3540 \pm 1320$ | $3540 \pm 500$ |
| OPAIR | $\mathbf{3237 \pm 180}$ | $\mathbf{7223 \pm 151}$ | $\mathbf{7613 \pm 604}$ | $\mathbf{3789 \pm 243}$ | $\mathbf{7064 \pm 42}$ | $\mathbf{5239 \pm 184}$ | $\mathbf{5381 \pm 32}$ |

**Paired repopulation.** The two pairing controls isolate the two sides of Equation 8. OPAIR-Zero sets both outgoing vectors to zero, preserving the network function without a learning signal, and OPAIR-Same sets both to $+\alpha v_d^\ell$, giving a learning signal but adding $2\alpha h_d^\ell v_d^\ell$ to the next layer. OPAIR-Same outperforms OPAIR-Zero on six of the seven tasks and matches it on Humanoid, and OPAIR outperforms both on all seven, including 7613 versus 2080 and 3656 on DemonAttack (Table 3). The learning signal thus helps on its own, and combining it with function preservation helps most. Since same-sign copies receive identical updates, the gap between OPAIR and OPAIR-Same also reflects the opposite updates of the two copies. Appendix B.5 details the controls.

**Reset budget.** OPAIR also resets far fewer neurons than the baselines, and ReDo accumulates 136–416 times as many resets as OPAIR over training (Table 5 in Appendix B.4). To exclude the number of reset neurons as an explanation of the performance gap, we cap ReDo and ReGraMa at 2% of the neurons in each layer per recycling event, which limits their resets while keeping recycling active. Relative to the uncapped runs in Tables 1 and 2, the cap raises ReDo on Ant and Walker2d, lowers it on all four Atari games, and lowers ReGraMa on six of the seven tasks. OPAIR achieves the highest mean on all seven tasks against both capped baselines (Table 3), so its advantage stems from how it repopulates released capacity rather than from resetting fewer neurons.

### 4.5 HYPERPARAMETER ABLATIONS

We vary $\tau$, which controls low-activity capacity release, and $\alpha$, which scales the paired outgoing weights and their gradient signals at repopulation (Table 4). OPAIR is robust to the threshold. Apart from Asterix under conservative release, where the return drops from 3237 to 2680, every non-default threshold stays within 3% of the default mean, and the higher threshold is marginally better on Seaquest and Ant. The default scale gives the highest mean on Asterix, Seaquest, and Walker2d, and on Ant both non-default scales exceed the default by at most 3%. Appendix B.6 provides the full $\tau$ and $\alpha$ curves and an additional recycling-period ablation.

Table 4: Sensitivity to (a) the activity threshold $\tau$ and (b) zero-pair scale $\alpha$. Entries are final return, mean $\pm$ standard deviation across five seeds, with the largest mean in bold. Thresholds (lower/default/higher) are $0.05/0.10/0.15$ on Atari and $0.005/0.02/0.05$ on MuJoCo; scales are $0.10/0.25/0.50$ in both benchmarks.

**(a) Activity threshold $\tau$**

| Task | Lower | Default | Higher |
|---|---|---|---|
| Asterix | 2679.7 $\pm$ 185.0 | **3237.1 $\pm$ 180.3** | 3217.0 $\pm$ 161.7 |
| Seaquest | 3700.4 $\pm$ 232.7 | 3788.9 $\pm$ 243.4 | **3792.9 $\pm$ 484.5** |
| Ant | 6869.6 $\pm$ 35.5 | 7064.4 $\pm$ 42.4 | **7181.4 $\pm$ 138.6** |
| Walker2d | 5185.2 $\pm$ 149.9 | **5238.7 $\pm$ 183.8** | 5158.3 $\pm$ 165.2 |

**(b) Zero-pair scale $\alpha$**

| Task | Lower | Default | Higher |
|---|---|---|---|
| Asterix | 3026.9 $\pm$ 128.6 | **3237.1 $\pm$ 180.3** | 2969.9 $\pm$ 174.1 |
| Seaquest | 3366.2 $\pm$ 295.9 | **3788.9 $\pm$ 243.4** | 3710.6 $\pm$ 391.6 |
| Ant | **7283.3 $\pm$ 81.7** | 7064.4 $\pm$ 42.4 | 7130.8 $\pm$ 107.1 |
| Walker2d | 5016.3 $\pm$ 235.1 | **5238.7 $\pm$ 183.8** | 5048.9 $\pm$ 273.7 |

## 5 Related Work

**Plasticity interventions in deep reinforcement learning.** Deep reinforcement learning agents gradually lose the ability to adapt as training changes their representations and optimization dynamics (Lyle et al., 2022; Kumar et al., 2020; Abbas et al., 2023). Periodic network or layer resets restore this ability under non-stationarity, primacy bias, and high replay ratios (Nikishin et al., 2022; D'Oro et al., 2022; Igl et al., 2020), and later work reduces reset-induced disruption through ensembles (Kim et al., 2023), cross-timescale knowledge transfer (Lee et al., 2024), or plasticity injection (Nikishin et al., 2023). DrM promotes active representations by minimizing the dormant ratio (Xu et al., 2024).

**Function-preserving capacity adaptation.** Network growth introduces trainable capacity while retaining learned computation. Net2Net widens and deepens trained networks through function-preserving transformations (Chen et al., 2015), Splitting Steepest Descent and Firefly split existing neurons by objective-dependent criteria (Wu et al., 2019; 2020), GradMax initializes new neurons to produce strong gradients while preserving the network function (Evci et al., 2022), and Neuroplastic Expansion grows high-gradient connections while pruning dormant neurons (Liu et al., 2025). These methods enlarge the model, and neuron recycling poses the corresponding fixed-capacity problem. Within this setting, zero-pair repopulation leaves the donor unchanged and gives its two copies opposite gradients, while the widening operator of Net2Net divides the outgoing weights among identical copies with identical gradients.

## 6 Conclusion

We revisit neuron recycling as a two-stage procedure of release and repopulation and assess both stages by the training objective. We prove that the activation-based release of ReDo causes only a small objective perturbation, so the effect of recycling is determined by repopulation, which we formalize as preserving the network function while maximizing the first-order objective reduction available to the recycled neurons along feature-rescaling directions. Based on this formalization, OPAIR replaces each pair of released neurons with two copies of an unreleased neuron with opposite outgoing weights, which preserves the network function while enabling the released capacity to learn from the objective, and chooses the copied neurons by the objective to maximize this criterion. Across 27 Atari and MuJoCo tasks, OPAIR achieves the highest mean final return on every task, and mechanism analyses confirm that both the paired construction and the objective-guided selection of copied neurons contribute.

## AI USE STATEMENT

Generative AI tools assisted with mathematical proof checking, code verification, and manuscript polish. They were not used to generate experimental data or results. The authors reviewed and verified all AI-assisted outputs and take full responsibility for the final content of the paper.

## REFERENCES

Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C Machado. Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*, pp. 620–636. PMLR, 2023.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Shibhansh Dohare, Richard S Sutton, and A Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop, NeurIPS*, 2022.

Utku Evci, Bart van Merriënboer, Thomas Unterthiner, Max Vladymyrov, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information. *arXiv preprint arXiv:2201.05125*, 2022.

Vivek F Farias and Adam D Jozefiak. Self-normalized resets for plasticity in continual learning. *arXiv preprint arXiv:2410.20098*, 2024.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

Woojun Kim, Yongjae Shin, Jongeui Park, and Youngchul Sung. Sample-efficient and safe deep reinforcement learning via reset deep ensemble agents. *Advances in Neural Information Processing Systems*, 36:53239–53260, 2023.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

Hojoon Lee, Hyeonseo Cho, Hyunseung Kim, Donghu Kim, Dugki Min, Jaegul Choo, and Clare Lyle. Slow and steady wins the race: Maintaining plasticity with hare and tortoise networks. *arXiv preprint arXiv:2406.02596*, 2024.

Jiashun Liu, Johan S Obando Ceron, Aaron Courville, and Ling Pan. Neuroplastic expansion in deep reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 90056–90078, 2025.

Jiashun Liu, Zihao Wu, Johan Obando Ceron, Pablo Samuel Castro, Aaron Courville, and Ling Pan. Measure gradients, not activations! enhancing neuronal activity in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 38:79546–79571, 2026.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*, 2022.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36:37142–37159, 2023.

Haoyuan Qin, Chennan Ma, Mian Deng, Zhengzhu Liu, Songzhu Mei, Xinwang Liu, Cheng Wang, and Siqi Shen. The dormant neuron phenomenon in multi-agent reinforcement learning value factorization. *Advances in Neural Information Processing Systems*, 37:35727–35759, 2024.

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.

Lemeng Wu, Dilin Wang, and Qiang Liu. Splitting steepest descent for growing neural architectures. *Advances in Neural Information Processing Systems*, 32, 2019.

Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly neural architecture descent: A general approach for growing neural networks. *Advances in Neural Information Processing Systems*, 33: 22373–22383, 2020.

Guowei Xu, Ruijie Zheng, Yongyuan Liang, Xiyao Wang, Zhecheng Yuan, Tianying Ji, Yu Luo, Xiaoyu Liu, Jiaxin Yuan, Pu Hua, et al. Drm: Mastering visual reinforcement learning through dormant ratio minimization. In *International Conference on Learning Representations*, volume 2024, pp. 56219–56243, 2024.

# A Additional Theory and Proofs

## A.1 Proof of Proposition 1

Removing neuron $s$ changes the next-layer pre-activation by

$$\Delta z^{\ell+1}(x) = -h_s^\ell(x) v_s^\ell. \tag{14}$$

Hence

$$\left\|\Delta z^{\ell+1}(x)\right\| \le B_\ell |h_s^\ell(x)|. \tag{15}$$

Let $\Phi_\ell$ denote the downstream mapping from $z^{\ell+1}$ to the network output. Local Lipschitz continuity gives

$$\begin{aligned}
\left\|F_{\text{mask}}^{(s)}(x) - F_{\text{old}}(x)\right\| &= \left\|\Phi_\ell\left(z^{\ell+1} + \Delta z^{\ell+1}\right) - \Phi_\ell\left(z^{\ell+1}\right)\right\| && (16)\\
&\le \lambda_\ell \left\|\Delta z^{\ell+1}(x)\right\| && (17)\\
&\le \lambda_\ell B_\ell |h_s^\ell(x)|. && (18)
\end{aligned}$$

Applying the local $\kappa$-Lipschitz condition for the loss yields

$$\left|\mathcal{L}\left(F_{\text{mask}}^{(s)}(x)\right) - \mathcal{L}\left(F_{\text{old}}(x)\right)\right| \le \kappa \lambda_\ell B_\ell |h_s^\ell(x)|. \tag{19}$$

Taking expectation over $x \sim \mathcal{D}$ proves Equation 3.

## A.2 Release of Multiple Neurons and the Masked Network

For a set $S_\ell$ released from layer $\ell$, the change in the next-layer pre-activation is

$$\Delta z^{\ell+1}(x) = -\sum_{s \in S_\ell} h_s^\ell(x) v_s^\ell. \tag{20}$$

Under the assumptions of Proposition 1 and $\|v_s^\ell\| \le B_\ell$ for every selected neuron, the triangle inequality gives

$$\left\|\Delta z^{\ell+1}(x)\right\| \le B_\ell \sum_{s\in S_\ell} |h_s^\ell(x)|. \tag{21}$$

Applying the downstream and loss Lipschitz bounds and taking expectation yields

$$\left|\mathcal{J}(F_{\text{mask}}^{S_\ell}) - \mathcal{J}(F_{\text{old}})\right| \le \kappa\lambda_\ell B_\ell \sum_{s\in S_\ell} \mathbb{E}_{x\sim\mathcal{D}}\left[|h_s^\ell(x)|\right]. \tag{22}$$

For selected sets $S_\ell$ in each hidden layer, define the diagonal mask

$$M_\ell = I - \sum_{s\in S_\ell} e_s e_s^\top. \tag{23}$$

With $M_0 = I$ and $\bar{a}^0 = x$, the masked forward pass is

$$\bar{z}^\ell = W^{\ell,\text{old}} M_{\ell-1}\bar{a}^{\ell-1} + b^{\ell,\text{old}}, \qquad \bar{a}^\ell = \sigma(\bar{z}^\ell), \tag{24}$$

and the masked output is

$$F_{\text{mask}}(x) = W^{L+1,\text{old}} M_L \bar{a}^L + b^{L+1,\text{old}}. \tag{25}$$

Equation 22 bounds the release perturbation of a set of neurons when their activations are small.

The bound also extends to release across multiple layers by a telescoping argument. Consider masking layers from the output toward the input. When layer $\ell$ is masked, its upstream computation is unchanged, so its activations are those of $F_{\text{old}}$. Let $\widetilde{\lambda}_\ell$ bound the Lipschitz constant of the downstream mapping with the later-layer masks already applied. Under uniform loss and downstream Lipschitz bounds over these interventions and $\|v_s^\ell\| \le B_\ell$ for each selected neuron, applying the single-layer bound at each step and summing by the triangle inequality gives

$$|\mathcal{J}(F_{\text{mask}}) - \mathcal{J}(F_{\text{old}})| \le \kappa \sum_{\ell=1}^{L} \widetilde{\lambda}_\ell B_\ell \sum_{s\in S_\ell} \mathbb{E}_{x\sim\mathcal{D}}\left[|h_s^\ell(x)|\right]. \tag{26}$$

### A.3 Repopulation Perturbation

Recall $R_{\text{mask}}(q)$ from Equation 9. If the loss is locally $\kappa$-Lipschitz with respect to the network output, then

$$\begin{aligned} |\mathcal{J}(F_{\text{rec}}) - \mathcal{J}(F_{\text{mask}})| &\le \kappa\mathbb{E}_{x\sim\mathcal{D}}\left[\|F_{\text{rec}}(x;q) - F_{\text{mask}}(x)\|_2\right] && (27)\\ &\le \kappa\sqrt{\mathbb{E}_{x\sim\mathcal{D}}\left[\|F_{\text{rec}}(x;q) - F_{\text{mask}}(x)\|_2^2\right]} && (28)\\ &= \kappa\sqrt{R_{\text{mask}}(q)}. && (29) \end{aligned}$$

The second inequality follows from Cauchy–Schwarz. The constraint $R_{\text{mask}}(q) = 0$ in Equation 8 therefore leaves $\mathcal{J}$ unchanged by repopulation. Combining the repopulation perturbation with the release perturbation of Stage I gives

$$|\mathcal{J}(F_{\text{rec}}) - \mathcal{J}(F_{\text{old}})| \le |\mathcal{J}(F_{\text{rec}}) - \mathcal{J}(F_{\text{mask}})| + |\mathcal{J}(F_{\text{mask}}) - \mathcal{J}(F_{\text{old}})|. \tag{30}$$

Thus the total objective change decomposes into a release perturbation and a repopulation perturbation.

### A.4 Learning Signal of a Repopulated Slot

This subsection justifies the first-order objective reduction in Equation 7. For each released slot $s \in S$ in layer $\ell$, introduce a rescaling coordinate $\beta_s$ that replaces the feature $h_s^\ell(x)$ by $(1+\beta_s)h_s^\ell(x)$ and leaves all other computations unchanged. Let $\boldsymbol{\beta} = (\beta_s)_{s\in S}$, let $\widetilde{\ell}_x(\boldsymbol{\beta})$ denote the resulting per-sample loss, and define $\widetilde{\mathcal{J}}_q(\boldsymbol{\beta}) = \mathbb{E}_{x\sim\mathcal{D}}[\widetilde{\ell}_x(\boldsymbol{\beta})]$ as in Equation 5. These coordinates serve only as an analytical device, and all derivatives are evaluated at $\boldsymbol{\beta} = 0$. Since $h_s^\ell$ affects the loss only through its contribution $h_s^\ell(x)v_s^\ell$ to the next-layer pre-activation, the chain rule gives

$$\left.\frac{\partial \widetilde{\ell}_x}{\partial \beta_s}\right|_{\boldsymbol{\beta}=0} = h_s^\ell(x)\,\frac{\partial \mathcal{L}}{\partial h_s^\ell(x)} = h_s^\ell(x)\,\left(v_s^\ell\right)^\top \delta^{\ell+1}(x). \tag{31}$$

Let $\psi_s$ contain the incoming weights and bias of slot $s$, and write its pre-activation as the inner product between $\psi_s$ and the corresponding bias-augmented input. If $\sigma$ is positively one-homogeneous, that is, $\sigma(cz) = c\,\sigma(z)$ for $c > 0$, then replacing $\psi_s$ by $(1+\beta_s)\psi_s$ with $1+\beta_s > 0$ rescales the feature by the same factor:

$$\sigma\big((1+\beta_s)z_s^\ell(x)\big) = (1+\beta_s)h_s^\ell(x). \tag{32}$$

For such activations, including ReLU, the rescaling coordinate is therefore a direction in parameter space. Taking the expectation of Equation 31 gives

$$\left.\frac{\partial \widetilde{\mathcal{J}}_q}{\partial \beta_s}\right|_{\boldsymbol{\beta}=0} = \mathbb{E}_{x\sim\mathcal{D}}\left[h_s^\ell(x)\frac{\partial \mathcal{L}}{\partial h_s^\ell(x)}\right] = \psi_s^\top \nabla_{\psi_s}\mathcal{J}. \tag{33}$$

Thus the objective-level rescaling gradient is the directional derivative of the training objective along the incoming parameters. For the gradient step $\boldsymbol{\beta}' = -\eta\nabla_{\boldsymbol{\beta}}\widetilde{\mathcal{J}}_q(0)$, a first-order expansion yields

$$\widetilde{\mathcal{J}}_q(0) - \widetilde{\mathcal{J}}_q(\boldsymbol{\beta}') = \eta\left\|\nabla_{\boldsymbol{\beta}}\widetilde{\mathcal{J}}_q(0)\right\|_2^2 + O(\eta^2) = \Delta(q) + O(\eta^2). \tag{34}$$

Appendix A.7 bounds the remainder for a zero-pair.

### A.5 Scale Dependence of the First-Order Objective Reduction

This subsection considers positively one-homogeneous activations such as ReLU and repopulations whose released slots contribute zero in total to the next-layer pre-activation, that is, $\sum_{s\in S} h_s^\ell(x)v_s^\ell = 0$ for every $x$. Zero-pair repopulation belongs to this class.

**Growth with parameter scales.** For $c, c' > 0$, let $q_{c,c'}$ replace the incoming parameters $\psi_s$ of every released slot by $c\,\psi_s$ and its outgoing vector $v_s^\ell$ by $c'v_s^\ell$. Positive homogeneity gives $h_s^\ell(x) \mapsto c\,h_s^\ell(x)$, so the total contribution becomes $cc'\sum_{s\in S} h_s^\ell(x)v_s^\ell = 0$. The next-layer pre-activation, the network function, and the gradient $\delta^{\ell+1}$ are therefore unchanged, the constraint in Equation 8 still holds, and each learning signal $h_s^\ell(x)(v_s^\ell)^\top\delta^{\ell+1}(x)$ scales by $cc'$. By Equation 7,

$$\Delta(q_{c,c'}) = (cc')^2\,\Delta(q). \tag{35}$$

Whenever a feasible repopulation has $\Delta(q) > 0$, the first-order objective reduction is thus unbounded over repopulations with unbounded scales, while the rescaled slots compute the same features up to scale. Bounded scales make the maximization in Equation 8 compare features rather than parameter magnitudes.

**Accuracy of the first-order approximation.** Let $\boldsymbol{\beta}$ denote the rescaling coordinates of Appendix A.4. Since the total contribution vanishes at $\boldsymbol{\beta} = 0$, the released slots enter the per-sample loss through

$$z^{\ell+1}(x;\boldsymbol{\beta}) = z^{\ell+1}_{\mathrm{mask}}(x) + \sum_{s\in S}\beta_s h_s^\ell(x)v_s^\ell, \tag{36}$$

where $z^{\ell+1}_{\mathrm{mask}}$ is the next-layer pre-activation of $F_{\mathrm{mask}}$. Let $H^{\ell+1}(x)$ denote the Hessian of the loss with respect to $z^{\ell+1}$ at $F_{\mathrm{mask}}$. The gradient $r_x$ and Hessian $G_x$ of $\widetilde{\ell}_x$ at $\boldsymbol{\beta} = 0$ have entries

$$(r_x)_s = h_s^\ell(x)(v_s^\ell)^\top\delta^{\ell+1}(x), \qquad (G_x)_{st} = h_s^\ell(x)h_t^\ell(x)(v_s^\ell)^\top H^{\ell+1}(x)v_t^\ell. \tag{37}$$

Let $r = \mathbb{E}_{x\sim\mathcal{D}}[r_x] = \nabla_{\boldsymbol{\beta}}\widetilde{\mathcal{J}}_q(0)$ and $G = \mathbb{E}_{x\sim\mathcal{D}}[G_x] = \nabla^2_{\boldsymbol{\beta}}\widetilde{\mathcal{J}}_q(0)$. Under the rescaling $q_{c,c'}$, $\delta^{\ell+1}$ and $H^{\ell+1}$ are unchanged, so $r$ scales by $cc'$ and $G$ by $(cc')^2$. For the objective-level gradient step $\boldsymbol{\beta}' = -\eta r$, a second-order expansion gives

$$\widetilde{\mathcal{J}}_q(0) - \widetilde{\mathcal{J}}_q(\boldsymbol{\beta}') = \eta\|r\|_2^2 - \frac{\eta^2}{2} r^\top G r + O(\eta^3). \tag{38}$$

The first-order term scales with $(cc')^2$ and the second-order term with $(cc')^4$, so their ratio scales with $\eta(cc')^2$. At a fixed step size, the first-order objective reduction $\Delta(q)$ therefore describes the actual objective reduction accurately only at moderate scales, which the bounds in $\mathcal{Q}$ maintain.

### A.6 Derivation of Equations 11 and 13

We first derive function preservation and the learning signals of a single zero-pair with general parameters $(\psi, v)$ as in Equation 10, sum over the pairs to obtain Equation 11, and then specialize to donor copies.

**Function preservation.** Both slots of a zero-pair share the incoming parameters $\psi$, so

$$h^\ell_{s_+}(x) = h^\ell_{s_-}(x) = h_\psi(x). \tag{39}$$

Their combined contribution to the next-layer pre-activation is

$$h^\ell_{s_+}(x)v^\ell_{s_+} + h^\ell_{s_-}(x)v^\ell_{s_-} = h_\psi(x)\,(+\alpha v) + h_\psi(x)\,(-\alpha v) = 0. \tag{40}$$

Every zero-pair in layer $\ell$ therefore contributes zero, and every next-layer pre-activation equals that of $F_{\text{mask}}$. Hence $F_{\text{rec}}(x) = F_{\text{mask}}(x)$ for every $x$, and $R_{\text{mask}} = 0$.

**Learning signals.** Since $F_{\text{rec}} = F_{\text{mask}}$, the gradient $\delta^{\ell+1}(x)$ at the next-layer pre-activation equals that of $F_{\text{mask}}$. Applying Equation 31 to the two slots gives

$$\left.\frac{\partial\widetilde{\ell}_x}{\partial\beta_{s_\pm}}\right|_{\boldsymbol{\beta}=0} = h_\psi(x)\,(\pm\alpha v)^\top\,\delta^{\ell+1}(x) = \pm\alpha\, h_\psi(x) g_v(x), \tag{41}$$

where $g_v(x) = v^\top\delta^{\ell+1}(x)$. Taking expectation over $x\sim\mathcal{D}$, the two rescaling-coordinate gradients of the training objective are

$$+\alpha\,\mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x)g_v(x)] \qquad \text{and} \qquad -\alpha\,\mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x)g_v(x)]. \tag{42}$$

Their squared magnitudes therefore sum to

$$(\alpha\mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x)g_v(x)])^2 + (-\alpha\mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x)g_v(x)])^2 = 2\alpha^2\left(\mathbb{E}_{x\sim\mathcal{D}}[h_\psi(x)g_v(x)]\right)^2. \tag{43}$$

Multiplying by $\eta$ gives the contribution of a single pair to $\Delta(q)$. Each released slot belongs to exactly one pair, so the sum over $S$ in Equation 7 splits over the pairs, which gives Equation 11. By Equation 4, the per-sample incoming gradients of the two slots are $\pm\alpha\, g_v(x)\,\sigma'(z_\psi(x))\,a^{\ell-1}(x)^\top$, which are opposite.

**Donor copies.** For a donor $d \in A_\ell$ with $(\psi, v) = (\psi_d, v^\ell_d)$, we have $h_\psi = h^\ell_d$. The donor affects the loss only through its contribution $h^\ell_d(x)v^\ell_d$ to the next-layer pre-activation, so $g_v(x) = (v^\ell_d)^\top\delta^{\ell+1}(x) = \partial\mathcal{L}/\partial h^\ell_d(x) = g^\ell_d(x)$. Each pair therefore has objective sensitivity $u(\psi_d, v^\ell_d) = u^\ell_d$, and Equation 11 becomes

$$\Delta(q_D) = 2\eta\alpha^2\sum_{d\in D} u^\ell_d. \tag{44}$$

Every donor satisfies $\|\psi_d\| \le C_\ell$ and $\|v^\ell_d\| \le B_\ell$, so $q_D \in \mathcal{Q}$.

**Optimality of TopN.** Order the unreleased neurons so that $u^\ell_{(1)} \ge u^\ell_{(2)} \ge \cdots$. For any $D \subseteq A_\ell$ with $|D| = N$, the $k$-th largest value of $u^\ell_d$ over $D$ is at most $u^\ell_{(k)}$, so

$$\sum_{d\in D} u^\ell_d \le \sum_{k=1}^{N} u^\ell_{(k)} = \sum_{d\in \mathrm{TopN}_{d\in A_\ell}\, u^\ell_d} u^\ell_d. \tag{45}$$

By Equation 44, $\Delta(q_D)$ is a positive multiple of $\sum_{d\in D} u^\ell_d$, so $D_\ell = \mathrm{TopN}_{d\in A_\ell}\, u^\ell_d$ maximizes $\Delta(q_D)$, with ties broken arbitrarily, which establishes Equation 13.

**Multiple layers.** For the feed-forward network defined in Section 2.1, the function-preservation result extends to simultaneous recycling in multiple layers when each released slot belongs to exactly one pair and every donor is unreleased. Copying rows in a later layer preserves the opposite-column relation of an existing pair, while replacing columns in an earlier layer preserves the identical-row relation of a copied pair. These pairwise identities are maintained across all layerwise assignments. By induction over layers, the unreleased computation agrees with $F_{\mathrm{mask}}$ and every zero-pair contributes zero to the next-layer pre-activation. The final output therefore satisfies $F_{\mathrm{rec}}(x) = F_{\mathrm{mask}}(x)$.

### A.7 Objective Reduction of a Zero-Pair

For a donor $d$, let $\boldsymbol{\beta} = (\beta_+, \beta_-)^\top$ denote the rescaling coordinates of its two copies (Appendix A.4), and let $\widetilde{\mathcal{J}}_d(\boldsymbol{\beta})$ denote the resulting training objective. Consider a gradient update along these coordinates and the resulting objective reduction:

$$\boldsymbol{\beta}' = -\eta \nabla_{\boldsymbol{\beta}} \widetilde{\mathcal{J}}_d(0), \qquad \Delta_d(\eta) = \widetilde{\mathcal{J}}_d(0) - \widetilde{\mathcal{J}}_d(\boldsymbol{\beta}'). \tag{46}$$

**Proposition 2** (Objective sensitivity and objective reduction)**.** *Suppose $\widetilde{\mathcal{J}}_d$ has an $H$-Lipschitz gradient over the relevant neighborhood. Then*

$$2\eta\alpha^2 \left(1 - \frac{H\eta}{2}\right) u_d^\ell \le \Delta_d(\eta) \le 2\eta\alpha^2 \left(1 + \frac{H\eta}{2}\right) u_d^\ell. \tag{47}$$

*For $0 < \eta < 2/H$, the lower bound is positive. When $H\eta \ll 1$, the leading term is $2\eta\alpha^2 u_d^\ell$.*

**Proof.** Abbreviate

$$r = \nabla_{\boldsymbol{\beta}} \widetilde{\mathcal{J}}_d(0). \tag{48}$$

The update in Equation 46 is $\boldsymbol{\beta}' = -\eta r$. An $H$-Lipschitz gradient gives the two-sided first-order remainder bound

$$\left| \widetilde{\mathcal{J}}_d(\boldsymbol{\beta}') - \widetilde{\mathcal{J}}_d(0) - r^\top \boldsymbol{\beta}' \right| \le \frac{H}{2} \left\| \boldsymbol{\beta}' \right\|_2^2. \tag{49}$$

Substituting $\boldsymbol{\beta}' = -\eta r$ gives

$$\eta \left(1 - \frac{H\eta}{2}\right) \|r\|_2^2 \le \widetilde{\mathcal{J}}_d(0) - \widetilde{\mathcal{J}}_d(\boldsymbol{\beta}') \le \eta \left(1 + \frac{H\eta}{2}\right) \|r\|_2^2. \tag{50}$$

From Appendix A.6, $\|r\|_2^2 = 2\alpha^2 u_d^\ell$. Therefore,

$$\Delta_d(\eta) \ge \eta \left(1 - \frac{H\eta}{2}\right) \|r\|_2^2 \tag{51}$$

$$= 2\eta\alpha^2 \left(1 - \frac{H\eta}{2}\right) u_d^\ell, \tag{52}$$

and similarly,

$$\Delta_d(\eta) \le 2\eta\alpha^2 \left(1 + \frac{H\eta}{2}\right) u_d^\ell. \tag{53}$$

These bounds prove Proposition 2. For $H\eta \ll 1$, both bounds have the leading term $2\eta\alpha^2 u_d^\ell$, which is the contribution of a single zero-pair to Equation 11 with $(\psi, v) = (\psi_d, v_d^\ell)$.

## B Experimental Details and Additional Results

### B.1 Benchmarks, Networks, and Training Protocol

**Atari-17.** The games are Asterix, BeamRider, Breakout, DemonAttack, DoubleDunk, Enduro, IceHockey, Jamesbond, MsPacman, Pong, Qbert, RoadRunner, Seaquest, SpaceInvaders, WizardOfWor, YarsRevenge, and Zaxxon. All use the corresponding `ALE/*-v5` environment with the minimal action set. Observations comprise four stacked $84 \times 84$ grayscale frames, with max pooling and action repeat $4$. Sticky actions have probability $0.25$; random no-ops and FIRE reset are disabled. The episode limit is $27{,}000$ agent steps. Training rewards are clipped, and evaluation uses raw episode returns.

The DQN network has three ReLU convolutional layers with $32/64/64$ channels, kernel sizes $8/4/3$, and strides $4/2/1$, followed by a 512-unit ReLU layer and a linear action-value head. Inputs are divided by 255. Training uses 64 parallel environments, replay capacity $10^6$, batch size 32, discount 0.99, and Adam with learning rate $6.25 \times 10^{-5}$ and $\epsilon = 1.5 \times 10^{-4}$. Learning starts after 20,000 transitions; one optimizer update is scheduled per subsequent transition. The target-update period is 2000 transitions. Exploration decays from 1 to 0.01 over 250,000 transitions after warm-up. The budget of $2.5 \times 10^6$ transitions is aggregated across environments, corresponding to $10^7$ emulator frames.

Evaluation is scheduled every 250,000 transitions, yielding ten checkpoints. Each evaluation uses 20 environments, $\epsilon$-greedy actions with $\epsilon = 0.001$, and at least 125,000 evaluation transitions. Once that budget is reached, active episodes are allowed to finish. The per-run score is the mean raw return across completed episodes, so the episode count varies by game and checkpoint. Vectorized collection can move the logged step up to 63 transitions beyond the nominal checkpoint.

**MuJoCo-10.** Tasks are `Ant-v4`, `HalfCheetah-v4`, `Hopper-v4`, `Humanoid-v4`, `HumanoidStandup-v4`, `InvertedDoublePendulum-v4`, `Pusher-v5`, `Reacher-v4`, `Swimmer-v4`, and `Walker2d-v4`, accessed through Gymnasium. SAC uses BRO-Net actor and twin-critic architectures of width 256. Each network has an input dense layer followed by ReLU; the actor has one residual block and each critic has two. Each block contains two dense layers, with ReLU after the first layer and a residual connection around the block. The actor parameterizes a squashed Gaussian policy.

Training uses five parallel environments, replay capacity $10^6$, batch size 256, and 5000 initial transitions before learning. The replay ratio is 2, and the actor-update period is 2 critic updates. The discount is 0.99, the target-network averaging coefficient is 0.005, and the actor, critic, and entropy-temperature learning rates are $3 \times 10^{-4}$, $10^{-3}$, and $3 \times 10^{-4}$, respectively. The initial entropy temperature is 1, with an automatic target entropy based on action dimension. Each run collects $10^6$ transitions. Evaluation uses the deterministic policy for 20 episodes every 50,000 transitions, yielding twenty checkpoints.

**Recycling configurations.** Base trains the same agent without neuron recycling. All recycling methods use a default period of 1000 optimizer updates. On Atari, ReDo uses activity threshold 0.10 and scoring batch size 64; ReGraMa uses normalized gradient-magnitude threshold 0.01 and scoring batch size 128. Both reinitialize incoming weights and set outgoing weights to zero. OPAIR uses scoring batch size 128, activity threshold 0.10, and zero-pair scale 0.25, and copies each donor's incoming weights and biases exactly into both released slots. For all recycling methods, convolutional channels and the hidden fully connected layer are eligible, and output units are excluded.

On MuJoCo, all scoring batches contain 128 transitions. ReDo uses activity threshold 0.02, ReGraMa uses normalized gradient-magnitude threshold 0.01, and OPAIR uses activity threshold 0.02 and zero-pair scale 0.25. Recycling acts on the critics, and actor recycling is disabled. All recycling methods operate on the first dense layer inside each critic residual block, whose activations feed a common outgoing dense map. Modified Adam moments are cleared. The methods therefore share the base architecture and recycling locations, and differ only in their release and repopulation rules.

**Implementation of the two-stage construction.** OPAIR computes normalized mean absolute activation and selects the $2N_\ell$ lowest-activity neurons satisfying the activity threshold as released slots. Normalization divides all scores in a layer by a common factor, so a neuron with normalized score below $\tau$ satisfies $\mathbb{E}_{x\sim\mathcal{D}}[|h_s^\ell(x)|] < \tau$ times the mean activation of its layer, and Proposition 1 applies directly to the released neurons. Objective sensitivities are then computed for the unreleased neurons, and the $N_\ell$ neurons with the largest sensitivities are selected as donors. Each donor is assigned to two released slots through the zero-pair construction described in Algorithm 1.

For a convolutional channel, OPAIR first sums the activation–gradient product $hg$ over spatial locations for each sample, averages the resulting scalar across the scoring minibatch, and squares the batch mean. For a dense-layer neuron, objective sensitivity is estimated as the square of the minibatch mean of $hg$. These are plug-in minibatch estimates of Equation 12; they are not claimed to be unbiased. OPAIR normalizes the resulting objective-sensitivity scores by their layer mean, preserving their ranking.

### B.2 Atari Learning Curves

Figure 2 gives the complete recorded trajectory for each Atari game, using the same five random seeds as Table 1.

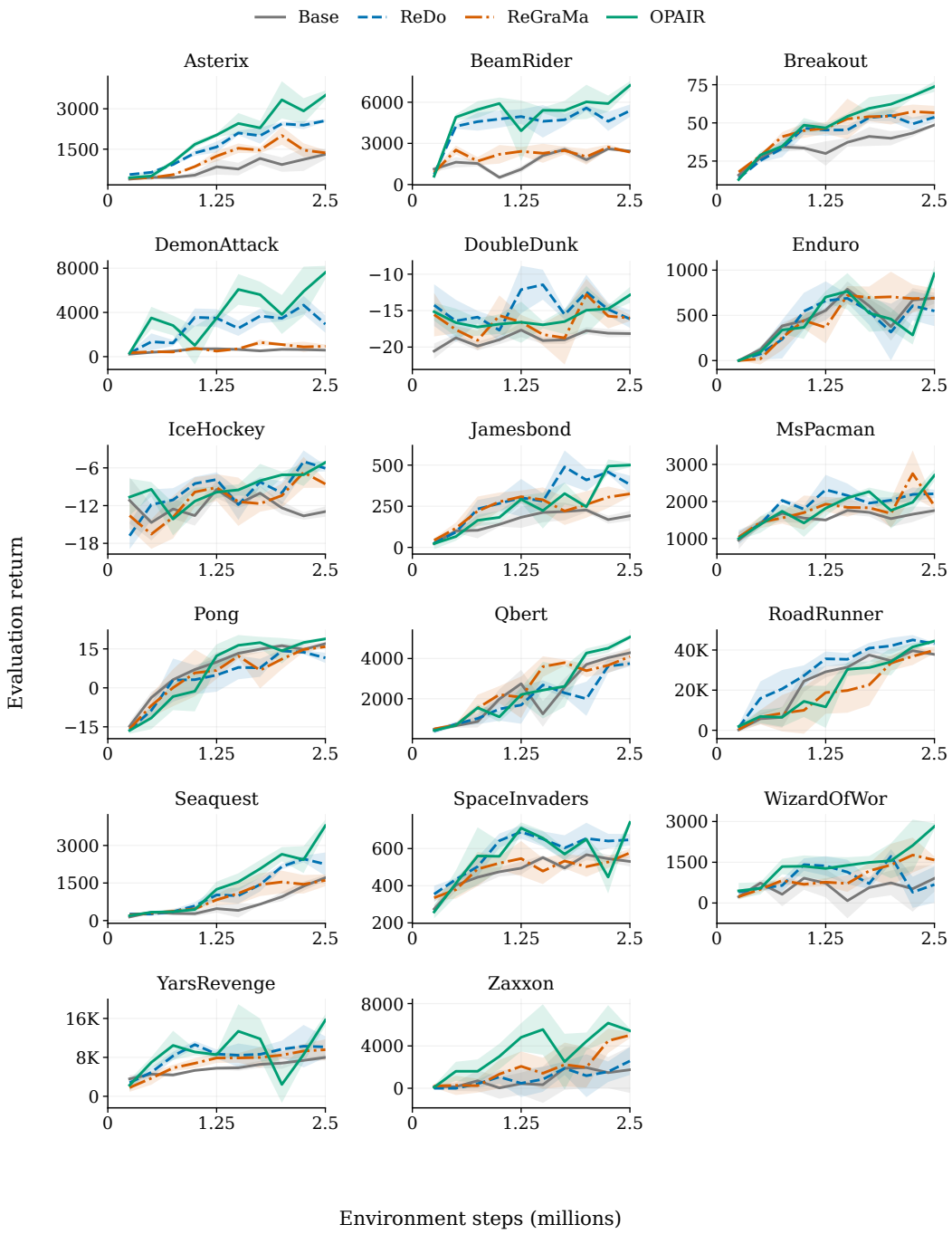



Figure 2: Atari-17 learning curves. Lines and shading show the mean ± standard deviation across five seeds. The horizontal axis counts environment transitions after action repeat; 2.5M transitions correspond to 10M emulator frames. Each panel uses its own return scale.

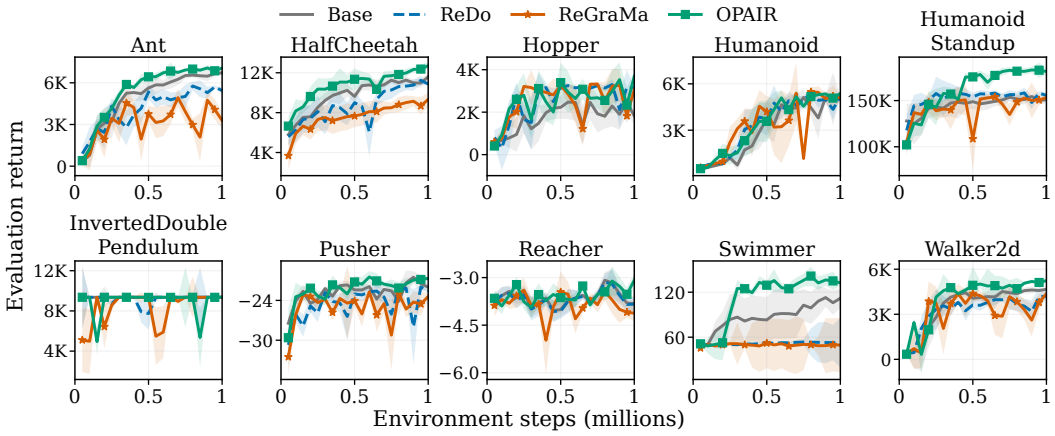



Figure 3: MuJoCo-10 learning curves with SAC and BRO-Net critics. Lines and shading show the mean ± standard deviation across five seeds.

### B.3 MuJoCo Learning Curves

### B.4 Re-reset Statistics

For each recycled neuron, we record every recycling event at which it is released. The re-reset interval of a neuron is the number of recycling events between two consecutive releases. We compute the mean interval of each neuron, average it over the neurons within each seed, and then average across the five seeds. Larger intervals indicate that recycled neurons take longer to return to a low-activity state. Cumulative resets sum the number of neurons reset at each recycling event over training, so a neuron that is reset repeatedly is counted each time. Table 5 reports both statistics for ReDo, OPAIR-Zero, and OPAIR on Asterix, Seaquest, Ant, and Walker2d.

Table 5: Re-reset statistics, averaged across five seeds. Bold marks the largest interval and the fewest cumulative resets.

| Task | Method | Mean Re-reset Interval | Cumulative Resets |
|---|---|---|---|
| Asterix | ReDo | 5.79 | 239,956.0 |
| Asterix | OPAIR-Zero | 309.13 | 3,627.0 |
| Asterix | OPAIR | **507.67** | **1,761.2** |
| Seaquest | ReDo | 4.34 | 264,748.8 |
| Seaquest | OPAIR-Zero | 189.04 | 4,783.5 |
| Seaquest | OPAIR | **335.32** | **1,404.5** |
| Ant | ReDo | 3.43 | 656,198.4 |
| Ant | OPAIR-Zero | 256.50 | 2,773.5 |
| Ant | OPAIR | **317.07** | **1,576.5** |
| Walker2d | ReDo | 2.11 | 1,190,457.0 |
| Walker2d | OPAIR-Zero | 101.15 | 11,565.6 |
| Walker2d | OPAIR | **149.03** | **5,065.0** |

### B.5 Mechanism Controls

**Donor controls.** The tasks are Asterix, BeamRider, DemonAttack, and Seaquest with DQN, and Ant, Walker2d, and Humanoid with SAC/BRO-Net. Random Donor samples distinct donors uniformly from the remaining neurons $A_\ell$ in the current layer. Activation Donor ranks them by mean absolute activation, and Gradient Donor ranks them by mean absolute activation-gradient magnitude. OPAIR ranks them by objective sensitivity. The variants share the activation-based release rule, number of recycled pairs, and zero-pair repopulation. Subsequent trajectories change the evolving activation and objective-sensitivity statistics, so the identity of recycled neurons can differ.

**OPAIR-Zero control.** OPAIR-Zero selects donors by objective sensitivity, copies each donor's incoming weights and biases into two released slots, and initializes both outgoing vectors to zero.

It therefore shares OPAIR's donor selection and initial activations, and isolates the effect of the oppositely signed outgoing vectors in Equation 10.

At initialization, zero outgoing weights remove the direct incoming-parameter gradient through that outgoing map. The nonzero pair supplies the learning signals induced by the current objective and characterized in Equation 11. Its effect on the subsequent optimization trajectory and return is measured empirically. Figure 4 shows the full return curves.

**OPAIR-Same control.** OPAIR-Same selects donors by objective sensitivity, copies each donor's incoming weights and biases into two released slots, and assigns both slots the outgoing vector $+\alpha v_d^\ell$. All other settings match OPAIR. The pair therefore adds $2\alpha h_d^\ell(x) v_d^\ell$ to the next-layer pre-activation, and its two copies receive identical gradients.

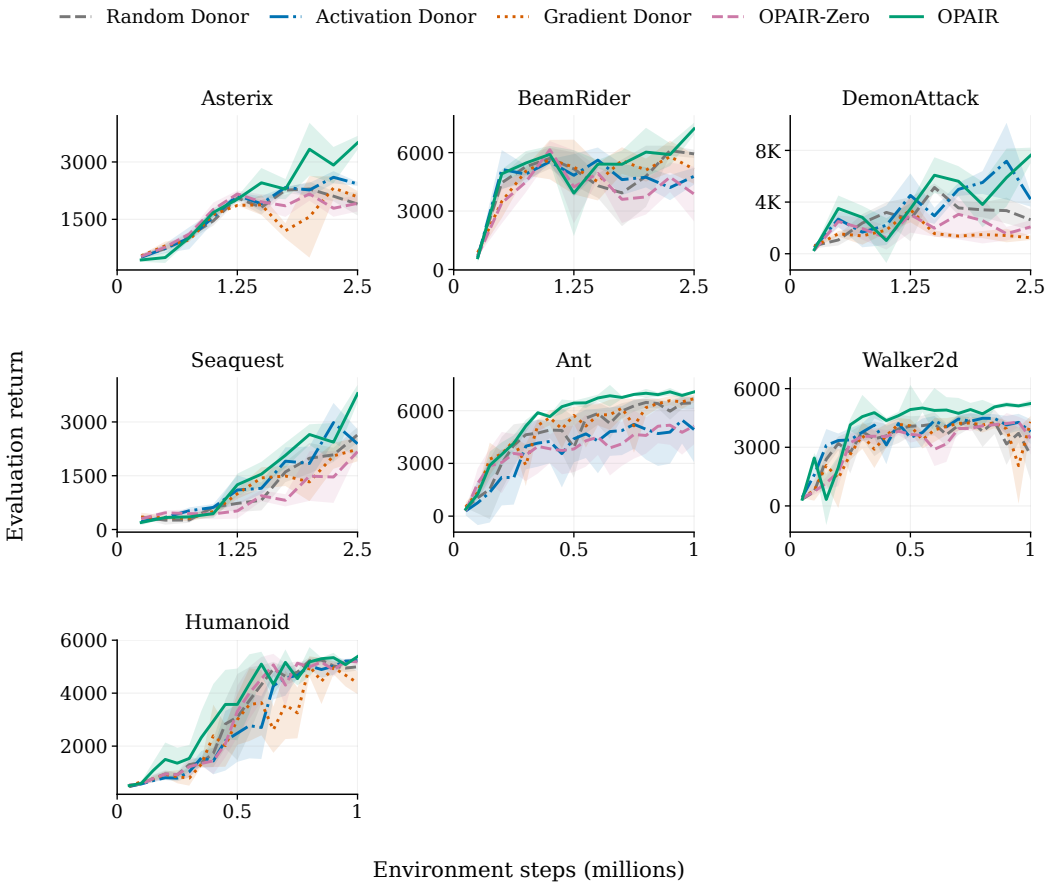



Figure 4: Mechanism learning curves for Table 3. Lines and shading show the mean ± standard deviation across five seeds. Atari and MuJoCo panels use their respective training budgets and raw returns.

### B.6 Additional Ablations

All ablations use Asterix, Seaquest, Ant, and Walker2d and change one configuration parameter relative to the corresponding main OPAIR configuration. Default curves and scores reuse the main runs.

**Activity threshold and zero-pair scale.** Thresholds are $0.05/0.10/0.15$ on Atari and $0.005/0.02/0.05$ on MuJoCo (Figure 5). Scales are $0.10/0.25/0.50$ for all four tasks (Figure 6). The threshold controls which low-activity neurons can be released. The scale changes nonzero outgoing magnitude and its initial backpropagated signal; direct cancellation within each zero-pair holds for every tested scale.

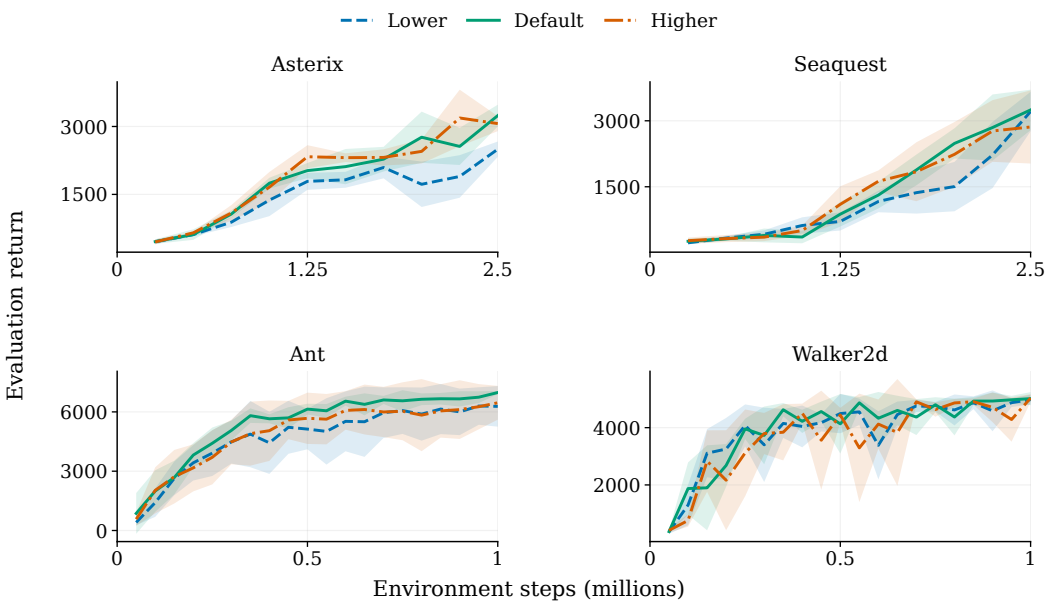



Figure 5: Activity-threshold ablation: mean ± standard deviation across five seeds. Lower/default/higher thresholds are $0.05/0.10/0.15$ on Atari and $0.005/0.02/0.05$ on MuJoCo.

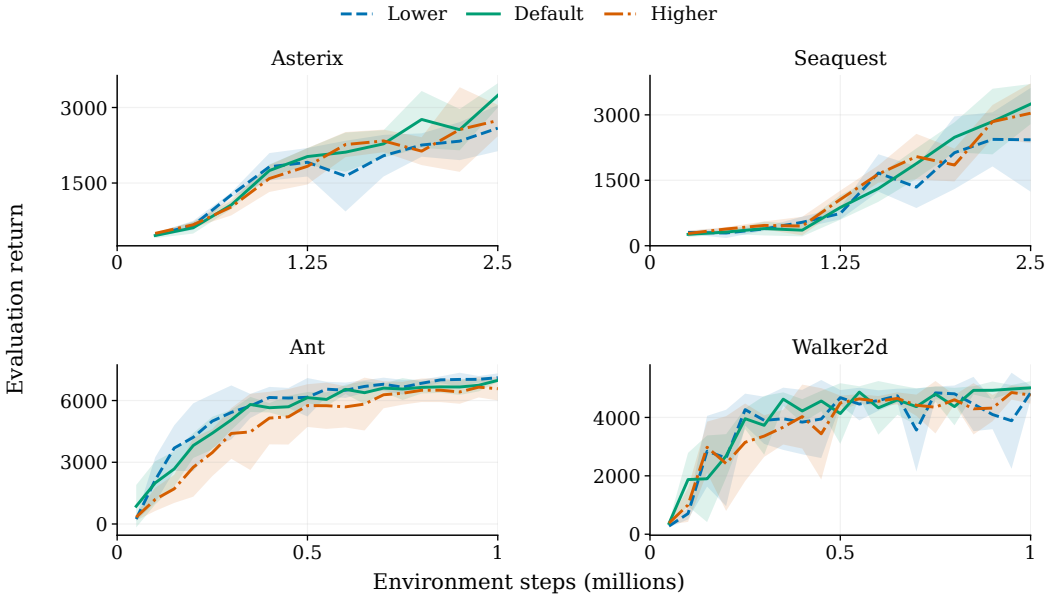



Figure 6: Zero-pair scale ablation: mean ± standard deviation across five seeds. Lower/default/higher scales are $0.10/0.25/0.50$ in both benchmarks.

**Recycling period.** Periods are $500/1000/5000$ optimizer updates in both benchmarks. The default gives the largest mean on all four tasks (Table 6). Period $5000$ reduces Ant's mean from $7064$ to $5642$, whereas Walker2d varies only between $4839$ and $5239$ across the tested periods. Figure 7 shows the trajectories, indicating that recycling frequency has task-dependent effects.

Table 6: Recycling-period ablation: final return, mean $\pm$ standard deviation across five seeds. Lower/default/higher periods are $500/1000/5000$ optimizer updates.

| Task | Lower | Default | Higher |
|---|---|---|---|
| Asterix | 2853.4 $\pm$ 370.6 | **3237.1 $\pm$ 180.3** | 2892.8 $\pm$ 608.7 |
| Seaquest | 2921.9 $\pm$ 707.7 | **3788.9 $\pm$ 243.4** | 3171.2 $\pm$ 807.2 |
| Ant | 6432.0 $\pm$ 1037.7 | **7064.4 $\pm$ 42.4** | 5642.0 $\pm$ 779.6 |
| Walker2d | 4839.4 $\pm$ 171.3 | **5238.7 $\pm$ 183.8** | 5071.8 $\pm$ 209.2 |

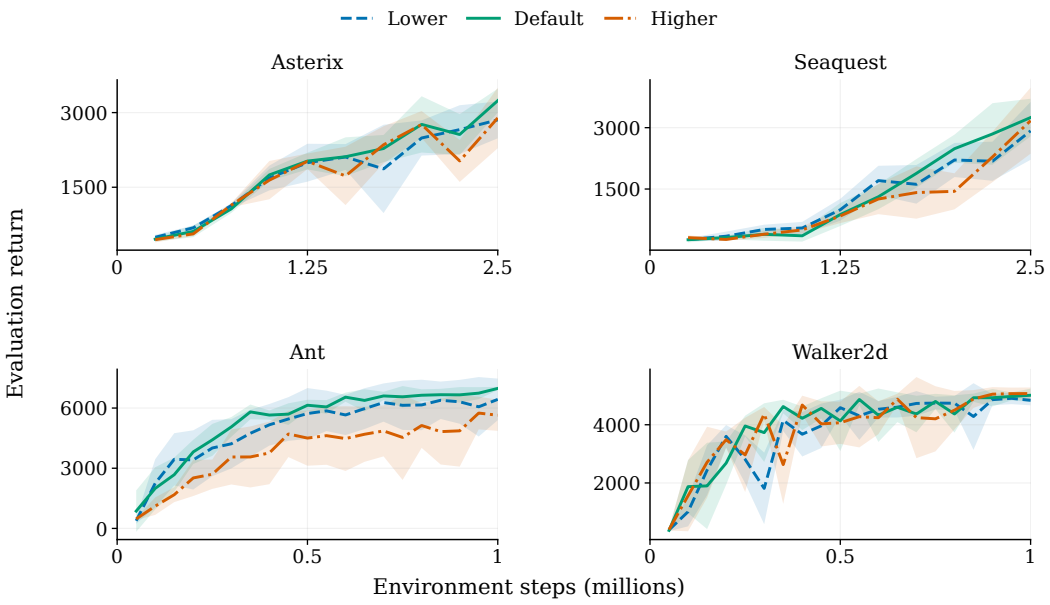



Figure 7: Recycling-period ablation: mean $\pm$ standard deviation across five seeds. Lower/default/higher settings are $500/1000/5000$ optimizer updates; the horizontal axis counts environment transitions.