SELF-SPARSIFYING REPRESENTATIONS IN JOINT-EMBEDDING PREDICTIVE ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Rectified LpJEPA enables direct control over representation sparsity in joint-embedding predictive architectures (JEPA), yet its performance degrades as representations approach extreme sparsity. We show that much of this degradation is an optimization cost rather than a limit of the sparse code, as aggressive sparsity shrinks gradients and limits early formation of informative features. We therefore introduce S^2-JEPA, a self-sparsifying JEPA that paces sparsification from loss curvature and corrects optimization strength from code density, adapting both throughout learning without labels or reference runs. At $\sim 96\%$ sparsity, reference-free S^2-JEPA improves frozen encoder and sparse projector accuracy over fixed-sparse training by up to 14.4 and 14.5 points, and exceeds its non-sparse counterpart on CIFAR-100 and STL-10, while retaining 97.2% of non-sparse accuracy on ImageNet-100. Its gains over fixed-sparse training hold across ResNet and ViT backbones and four optimizers. Pretrained on CIFAR-100, S^2-JEPA improves mean transfer accuracy across five datasets by 8.7 points over fixed-sparse training, matching non-sparse transfer, and outperforms dense JEPA codes in nearest-neighbor retrieval with $12\times$ less storage. Code will be publicly released.

1 INTRODUCTION

Sparse representations encode inputs using vectors with mostly zero coordinates. They reduce retrieval-index storage (Wen et al., 2025; Guo et al., 2026), allow inference to skip inactive computations (Kurtz et al., 2020; Mirzadeh et al., 2024), and make learned features easier to inspect (Huben et al., 2024; Gao et al., 2025). Sparse coding also models efficient information processing in sensory systems (Olshausen & Field, 1996). These benefits motivate learning sparse representations directly during self-supervised training.

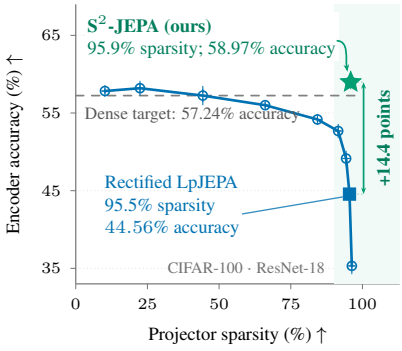

Figure 1: **S^2-JEPA recovers accuracy under high sparsity.** At comparable projector sparsity, it outperforms the fixed-target Rectified LpJEPA setting (square) and exceeds the dense target (dashed).

Joint-Embedding Predictive Architectures (JEPA) (Assran et al., 2023) learn by aligning embeddings of augmented views. LeJEPA (Balestriero & LeCun, 2025) prevents collapse through distribution matching, and CNN-JEPA (Kalapos & Gyires-Tóth, 2024) extends JEPAs to convolutional encoders. Rectified LpJEPA (Kuang et al., 2026) extends this approach to sparse representations using a rectified target whose mean controls the expected active fraction. Unlike penalty-based SparseJEPA (Hartman & Varshney, 2025), this sets the sparsity level directly, so training procedures can be compared at matched sparsity. However, its accuracy drops sharply near 95% sparsity, especially under a practical 100-epoch budget. On CIFAR-100, a fixed target at 95.5% sparsity loses 12.7 points of frozen encoder accuracy relative to the dense target. We call this drop the *sparsity cliff*.

Similar cliffs occur with activation penalties (Glorot et al., 2011; Song et al., 2025), top-k selection (Yang et al., 2021; Makhzani & Frey, 2014; Gao et al., 2025; Guo et al., 2026), and gradual weight sparsification (Zhu & Gupta, 2018; Evci et al., 2021; Hoefler et al., 2021). Prior work links this

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.

degradation to impaired gradient flow (Evci et al., 2022) and normalization-induced gradient skew (Adnan et al., 2026), leading to sparsity-aware parameterizations (Dey et al., 2024), learning-rate schedules (Liu et al., 2023), and gradual sparsification (Guo et al., 2026; Song et al., 2025). Yet it remains unclear whether the loss comes from the sparse code or from its training dynamics. Existing corrections also rely on proxy models, labeled validation data, or sweeps that must be repeated for each optimizer, backbone, or training budget.

We find that much of the cliff is an optimization cost that training can recover. With $10\times$ more updates, the fixed-target encoder reaches dense accuracy at 97.4% sparsity (65.1% vs. 65.3%), although projector accuracy remains 4.8 points lower. We identify two causes. First, sparse targets shrink gradients. Without optimizer-side rescaling, parameter movement falls to $0.18\times$ that of dense training. Second, high sparsity from the start restricts feature formation, causing an additional loss even at matched update magnitude. Both effects vary across training settings. Across various combinations of dataset, backbone, and optimizer, the encoder gap ranges from 1.7 to 12.7 points, and dense and sparse runs prefer different update magnitudes. A correction that improves CIFAR-100 encoder accuracy by 5.0 points reduces ImageNet-100 projector accuracy by 3.1 points when used throughout training. Thus, fixed corrections do not transfer reliably across settings.

We introduce S^2-JEPA, which adapts both corrections using measurements from the current run. Sparsity pacing lowers the target mean after loss curvature indicates adaptation, while magnitude correction scales the update magnitude from code density and remains active only during a movement shortfall. The method requires no labels or reference runs. At about 96% sparsity on CIFAR-100, it improves frozen encoder accuracy by 14.4 points over the fixed target, 14.5 points for the sparse projector code, and by 1.7 points over dense training (Fig. 1). Its frozen encoder matches or exceeds dense transfer accuracy on all five downstream datasets, its codes stay 95–96% sparse on those datasets, and with longer training it comes within 0.5 points of dense ImageNet-100 accuracy using $20\times$ fewer active coordinates.

Contributions.

- We show that the sparsity cliff in JEPA is largely an optimization cost, driven by density-dependent gradient shrinkage and restricted feature formation under early sparsification.
- We propose S^2-JEPA, which uses loss curvature to pace sparsification and code density to correct update magnitude, applying each only when needed without labels or reference runs.
- We evaluate S^2-JEPA across multiple datasets, backbones, optimizers, and training budgets. It improves on fixed-sparse Rectified LpJEPA in every setting, exceeds dense encoder accuracy on CIFAR-100 and STL-10, and remains competitive with dense training everywhere else. Its sparse codes retrieve better than dense JEPA codes, even when these are compressed to the same size.

2 Background

Rectified LpJEPA (Kuang et al., 2026) learns sparse, non-negative representations by rectifying the projector output and regularizing its distribution. Given two augmented views x and x' of the same input, it computes $z = \mathrm{ReLU}(h_\theta(x))$ and $z' = \mathrm{ReLU}(h_\theta(x'))$, where h_θ is the encoder followed by the projector and $z, z' \in \mathbb{R}^D_{\geq 0}$, with $D = 512$ (2048 on ImageNet-100). The ReLU activation allows individual coordinates to be exactly zero. The training objective combines an invariance loss with Rectified Distribution Matching Regularization,

$$L_\mu(\theta) = \alpha\,\mathbb{E}\|z - z'\|_2^2 + \lambda\,\mathbb{E}_c\Big[W_2^2\Big(P_{c^\top z}, P_{c^\top y_\mu}\Big) + W_2^2\Big(P_{c^\top z'}, P_{c^\top y_\mu}\Big)\Big], \qquad y_\mu \sim \textstyle\prod_{i=1}^D \mathcal{RGN}_p(\mu, \sigma). \tag{1}$$

where α and λ weight the two terms, c is sampled uniformly from the unit sphere, and W_2 denotes the 2-Wasserstein distance. Each coordinate of y_μ follows a rectified generalized Gaussian distribution, obtained by applying ReLU to a generalized Gaussian random variable with location μ and scale σ.

The probability that a target coordinate is nonzero determines its expected active fraction,

$$\frac{\mathbb{E}\|y_\mu\|_0}{D} = \Phi_{\mathcal{GN}_p(0,1)}\Big(\frac{\mu}{\sigma}\Big). \tag{2}$$

For fixed p and σ, μ controls target sparsity, allowing comparisons at the same target. Unless stated otherwise, we use $p = 1$ and $\sigma = \sigma_{\mathrm{GN}} = 1/\sqrt{2}$ following Kuang et al. (2026). At $\mu = 0$, the

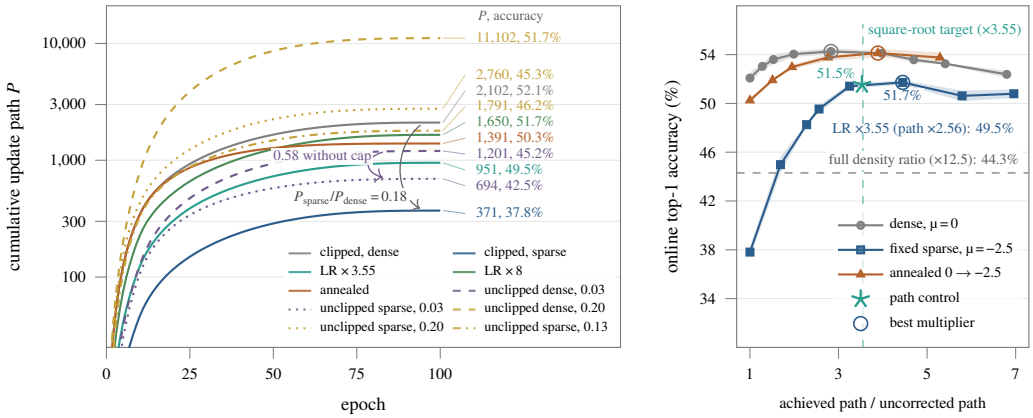

Figure 2: **Update magnitude of ResNet-18 on CIFAR-100.** (Left) Cumulative update paths over 100 epochs; endpoint labels show the final path and online encoder accuracy. (Right) Online encoder accuracy versus achieved path, normalized by each family's unscaled run. The star enforces the square-root target in closed loop; the dashed line shows full-density accuracy, whose path lies off-axis.

target has 50% zeros (which we call dense), while $\mu = -2.5$ gives about 98.5% zeros. Learned codes are denser, reaching 95.5% zeros on CIFAR-100 at $\mu = -2.5$. We report sparsity on learned projector outputs and denote their active fraction by f. We otherwise follow the optimization setup of Kuang et al. (2026) and use LARS (You et al., 2017), which scales each layer's update by its weight-to-gradient norm ratio.

3 Diagnosing the Sparsity Cliff

If the sparsity cliff is a limit of sparse codes, changing training while keeping the same final sparsity should not recover accuracy. We test two possible causes. Density-dependent gradient shrinkage may limit parameter updates, while early sparsification may restrict feature formation. We first measure the effect of update magnitude. We then compare fixed and annealed targets with matched total update paths to study sparsification timing.

3.1 Density-dependent gradient shrinkage

We begin with two Rectified LpJEPA models that differ only in their target mean. We use $\mu = 0$ for the dense reference and $\mu = -2.5$ for the fixed-sparse run. Under this setting, sparse training produces a 14.3-point deficit in online accuracy. To determine whether both models receive comparable updates, we measure the integrated update path $P = \sum_{k<K} \|\theta_{k+1} - \theta_k\|_2$ over the encoder and projector parameters, where k indexes optimization steps. Despite taking the same number of steps, the sparse run accumulates only $0.18\times$ the path of the dense run (Fig. 2, left). This large reduction in parameter movement suggests that update magnitude contributes to the sparsity cliff.

The effect of smaller gradients depends on how the optimizer rescales them. Optimizers that normalize each coordinate's update, such as AdamW (Loshchilov & Hutter, 2019), can compensate for changes in gradient scale. With AdamW, a ResNet-18 loses only 1.7 frozen encoder points under the same sparse target (§ 5.2). LARS (You et al., 2017) instead assigns each layer ℓ a trust ratio based on its weight and gradient norms. The implementation used by Kuang et al. (2026) also caps the layerwise learning rate (LR) at the base LR. For tensors receiving LARS scaling, the trust ratio and pre-momentum update direction are

$$\lambda_\ell = \eta \frac{\|w_\ell\|}{\|g_\ell\| + \gamma\|w_\ell\| + \epsilon} \qquad (3) \qquad\qquad u_\ell = \min\left(\frac{\lambda_\ell}{\mathrm{lr}}, 1\right)(g_\ell + \gamma w_\ell) \qquad (4)$$

where η denotes the trust coefficient, γ is the weight decay, and $\epsilon = 10^{-8}$. Momentum is then updated as $v_{\ell,k+1} = 0.9v_{\ell,k} + u_{\ell,k}$, producing the displacement $\Delta w_{\ell,k} = -\mathrm{lr}_k v_{\ell,k+1}$. Smaller gradient

norms increase λ_ℓ and should, in principle, allow LARS to compensate for changes in gradient scale. In our experiments, however, the cap remains active throughout training for every tensor receiving LARS scaling. The gradient norms also exceed the weight-decay term by approximately two orders of magnitude, yielding $u_\ell \approx g_\ell$. Smaller gradients therefore enter the momentum buffer without compensating rescaling.

We first test whether this reduction comes from the zero derivative of inactive ReLU coordinates. We keep hard ReLU in the forward pass but use the GELU derivative (Hendrycks & Gimpel, 2023) for backpropagation, allowing gradients through inactive coordinates. The path ratio and accuracy gap remain nearly unchanged at 0.17 and 15.6 points. Removing the LARS cap raises the ratio to 0.58 and narrows the gap, though part of this recovery at the original LR comes from lower dense accuracy because the rate was tuned for clipped LARS. After retuning with shared or separate rates, a gap of ~ 6 points remains. Thus, sparse-target performance depends on both the optimizer and sparse code.

We therefore keep LARS, the default in the Rectified LpJEPA recipe, and rescale its LR. Switching optimizers alone is not enough, as a fixed sparse target still costs 9.6 points with AdamW on ViT-S (§ 5.2). As a simple heuristic, if active coordinates contribute comparably to the squared gradient norm, the gradient norm scales with $\sqrt{f}$, which suggests the starting scale

$$\mathrm{lr} \longrightarrow m^\star \,\mathrm{lr}, \qquad m^\star = \sqrt{\frac{f_0}{f_\mu}}, \tag{5}$$

where f_0 and f_μ are the measured active fractions for the dense reference and target μ, respectively. At $\mu = -2.5$, this estimate gives $m^\star = 3.55$. Applying it as a fixed LR multiplier recovers 82% of the online accuracy gap. The update path increases by only $2.56\times$, however, because changing the LR also changes the optimization trajectory. Directly controlling the achieved path at approximately $3.5\times$ places accuracy within 0.2 points of the best fixed-sparse multiplier (Fig. 2, right), with the target determined from the measured density. In comparison, the full density ratio f_0/f_μ (Dey et al., 2024) specifies a substantially larger target and reduces accuracy by more than seven points. The appropriate rescaling also varies across runs. Dense and fixed-sparse training perform best at different path multipliers, so no single value suits both. The square-root rule therefore provides an initial scale that must be evaluated during training.

3.2 Preserving capacity during feature formation

Correcting update magnitude leaves a second possible cause. A fixed sparse target restricts the active coordinates available during early feature formation. We examine this effect by annealing the target mean from $\mu = 0$ to $\mu = -2.5$. Although the fixed and annealed runs reach similar final densities, annealing improves online accuracy by 12.5 points and also increases the update path. To separate these effects, we train both schedules at matched total paths of $P = 371$ and $P = 1390$. Increasing the path improves accuracy by 10.3 points on average, while annealing contributes a further 1.4–2.2 points at either path. Sparsification timing therefore affects accuracy beyond the final density and total parameter movement (more results in Tab. A8).

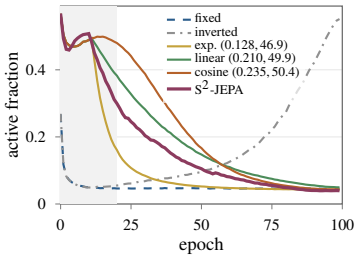

Figure 3: **Sparsification during training on CIFAR-100.** Tuples report the mean density and accuracy (%) for annealing schedules with the same final target. Shading marks the first 20 epochs. Curves averaged over 3 runs.

To study how coordinate availability affects learning, we compare exponential, linear, and cosine annealing schedules with the same final target (Fig. 3). Accuracy increases with the mean density observed during training, even though the final active fractions differ by at most 0.007. Interestingly, reversing the schedule produces lower accuracy despite ending at a substantially higher density. These results show that retaining active coordinates during feature formation matters and that final sparsity alone does not explain the observed accuracy differences.

This points to a natural next step: *can we combine sparsity scheduling with update-magnitude control?* Doing so by hand is expensive. The anneal shape and length and the LR multiplier come from sweeps, and the right choice, including whether to rescale the LR, changes with the optimizer and setting

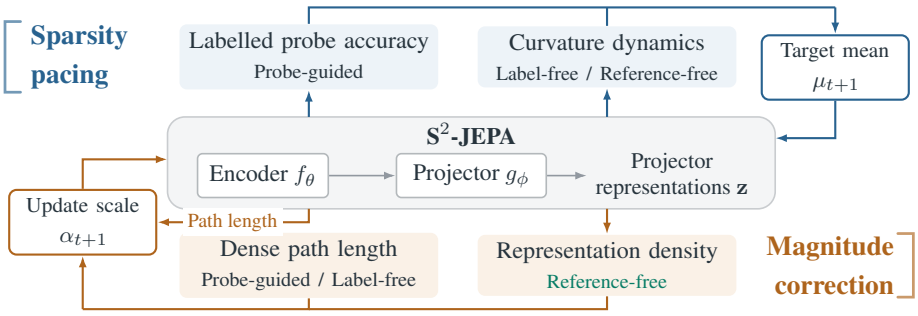

Figure 4: **Overview of S^2-JEPA.** Two feedback loops adapt the target and the update scale during training. *Sparsity pacing* (blue) sets the target mean μ_{t+1} using labelled probe accuracy or curvature dynamics, while *magnitude correction* (orange) sets the update scale α_{t+1} using dense path length or representation density. The three variants differ only in the signals they use.

(§ 5.2). This raises a harder question: *can the training run decide for itself when to sparsify and how far to scale its updates, without labels, a reference run, or tuning for each new setting?*

4 S^2-JEPA: Self-Steered Sparsity Training

To answer this question, we introduce S^2-JEPA, which measures both effects during training and adjusts each correction as the run progresses (Fig. 4). Sparsity pacing controls when to increase sparsity, while magnitude correction scales the updates. Both use only signals from the current run, without labels or a reference run. After a ten-epoch warm-up with $\mu = 0$ and $\alpha = 1$, the warm-up density serves as the reference. Measurements from each epoch set controls for the next.

4.1 Sparsity pacing from curvature dynamics

In § 3.2, we showed that accuracy depends on how long coordinates remain available during training, in addition to the final density. Sparsity should therefore increase at a rate that allows the model to adapt to each target. Abrupt target changes recover poorly (Tab. A8), so we decrease μ in small steps and allow training to respond before proceeding to the next target. Small steps also limit the change in the matching distribution between consecutive targets.

Proposition 1 (Continuity under target shifts)**.** *Let $\rho_\mu = \mathcal{RGN}_p(\mu, \sigma)$ with fixed $p > 0$ and $\sigma > 0$, and let $\Delta = |\mu - \mu'|$. Then $W_2(\rho_\mu, \rho_{\mu'}) \le \Delta$, the product distributions over D independent coordinates are within $\sqrt{D}\Delta$, and their projections along any unit vector c are within $\|c\|_1\Delta$. It bounds how much each step changes the matching target but does not by itself set the step size; App. A.3 gives the proof.*

To determine whether the model has adapted to its current target, we track the largest Hessian eigenvalue $\lambda_{\max,t}$ of the training loss, estimated once per epoch by power iteration. In our runs, this estimate generally decreases after each target change and can jump sharply after a large one (Fig. 5d). We therefore use its recent trend as a practical signal of whether sparsification can continue or the current target should be held. We define $q_t = \log \lambda_{\max,t}$, estimate its slope s_t over the previous five epochs, and compute the variability $\hat{\sigma}_t$ of its increments. The target is then updated as

$$\mu_{t+1} = \max\{\mu_{\min}, \mu_t - \delta_t\}, \qquad \delta_t = \begin{cases} 0, & |q_t - q_{t-1}| > 2\hat{\sigma}_t, \\ 0.05, & s_t < -0.02, \\ 0.025, & \text{otherwise.} \end{cases} \tag{6}$$

Here $\hat{\sigma}_t$ is the standard deviation of the last ten increments of q_t. A large curvature change in either direction holds μ fixed, while falling curvature permits a larger decrease. Because the rule depends on changes in $\log \lambda_{\max}$, its thresholds are independent of the overall curvature scale. We restrict μ to decrease monotonically, since allowing it to move back toward dense produces much denser final codes (App. D.2). Curvature controls the rate of sparsification, while $\mu_{\min}$ sets its endpoint,

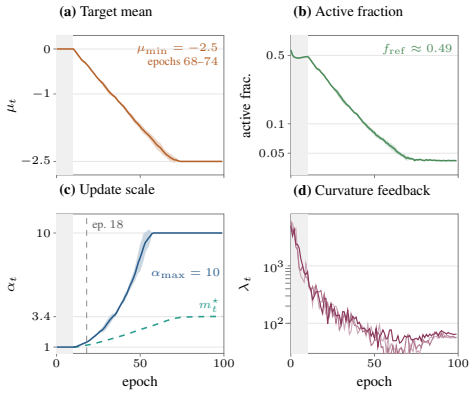

Figure 5: **S^2-JEPA on CIFAR-100.** After warm-up (grey), the target mean (a) falls in small steps while curvature (d) keeps falling, and the codes reach about 4% active (b). The update scale (c) grows past the square-root factor $m_t^\star$ (dashed) to its cap; the vertical line marks the second eligibility check. Bands: variation across runs.

Algorithm 1 S^2-JEPA.

Initialize $\mu_1 = 0$, $\alpha_1 = 1$, and flag $b = \text{false}$
for each epoch $t = 1, \ldots, T$ **do**
  Train with μ_t and α_t; measure curvature, movement, and active fraction
  if $t < 10$ **then**
    $\mu_{t+1} \leftarrow 0$; $\alpha_{t+1} \leftarrow 1$
  else
    if $t = 10$ **then**
      Fix f_{ref}; $b \leftarrow$ update scaling saturated
    end if
    Update μ_{t+1} using Equation (6)
    if $t = 18$ and b **then**
      $b \leftarrow$ movement shortfall present
    end if
    if b **then**
      Set $m_t^\star = \sqrt{f_{\text{ref}}/f_t}$; update r_t (§ 4.2)
      Update α_{t+1} using Equation (7)
    else
      $\alpha_{t+1} \leftarrow 1$
    end if
  end if
end for

analogous to k in top-k selection. We use $\mu_{\min} = -2.5$ in general and -3 on ImageNet-100. All other constants are shared across settings and were fixed before the reported runs (App. A.1).

4.2 Magnitude correction from representation density

In § 3.1, we estimated the missing update magnitude from density using $m^\star = \sqrt{f_0/f_\mu}$, where f_0 was obtained from a separate dense run. The warm-up allows S^2-JEPA to estimate this reference within the current run. We record its final active fraction as f_{ref}, track the current active fraction using an exponential moving average f_t, and compute $m_t^\star = \sqrt{\frac{f_{\text{ref}}}{f_t}}$.

The density-derived factor provides an initial scale rather than the optimal correction. As shown in § 3.1, applying 3.55 as a fixed multiplier increased the update path by only $2.56\times$ because changing the LR also alters the optimization trajectory. We therefore allow the correction to grow beyond $m_t^\star$ through a ramp factor r_t, initialized at 1. Each correction defines a target for the next epoch's parameter movement (App. A.2). The ramp increases by 5% per epoch, up to a maximum of 6, while training remains stable. Stability requires curvature to have been flat or decreasing over the previous five epochs, no curvature spike, and at least 90% of the previous movement target to have been reached. If any condition is unmet, the ramp remains fixed. The update scale α_t multiplies the base LR and approaches $r_t m_t^\star$ through the damped update

$$\alpha_{t+1} = \min\left\{\alpha_{\max},\, \alpha_t^{1-\beta}(r_t m_t^\star)^\beta\right\}, \qquad \beta = 0.7, \quad \alpha_{\max} = 10. \tag{7}$$

In log space, this update moves α_t a fraction β toward its target. The damping prevents short-term changes in f_t from producing abrupt changes in the LR, while $\alpha_{\max}$ bounds the applied multiplier.

When to correct. Smaller gradients reduce parameter movement only when the optimizer does not compensate for their scale (§ 3.1). Two eligibility checks therefore control magnitude correction. At the end of warm-up, the first activates correction only if layerwise update scaling is saturated for at least 95% of updates. Otherwise, as with AdamW, α remains at 1 and only sparsity pacing is used. After a short adaptation period, the second retains correction only while movement remains below its targets and continues to decrease. If either condition fails, α returns to 1 for the remainder of training, while sparsity pacing continues. We evaluate both checks in § 5.3. Algorithm 1 summarizes S^2-JEPA, and Fig. 5 shows trends across three CIFAR-100 runs. The target reaches its lower bound between epochs 68 and 74, while the update scale grows beyond the square-root factor. A linear anneal tuned to this trajectory performs similarly to S^2-JEPA on CIFAR-100 but falls about 3 projector points behind on ImageNet-100 (App. D.2).

Table 1: **Accuracy and representation sparsity on CIFAR-100 and ImageNet-100.** We report frozen linear-probe accuracy (%) for encoder and projector representations. Sparsity is the percentage of zero projector entries on evaluation images. Non-neg.: nonnegative self-supervised baselines; Rect.: Rectified LpJEPA. Dense: Rect. with $\mu = 0$. Rectified LpJEPA and S^2-JEPA use $p = 1$.

				CIFAR-100 / ResNet-18			ImageNet-100 / ResNet-50		
	Method	Label feedback	External reference	Sparsity (%) ↑	Encoder ↑	Projector ↑	Sparsity (%) ↑	Encoder ↑	Projector ↑
Dense	SimCLR	×	×	$0.00_{\pm 0.00}$	$60.70_{\pm 0.11}$	$54.96_{\pm 0.08}$	$0.00_{\pm 0.00}$	$75.02_{\pm 0.12}$	$69.84_{\pm 0.15}$
	VICReg	×	×	$0.00_{\pm 0.00}$	$63.04_{\pm 0.08}$	$55.64_{\pm 0.27}$	$0.00_{\pm 0.00}$	$79.08_{\pm 0.09}$	$70.50_{\pm 0.21}$
	LeJEPA	×	×	$0.00_{\pm 0.00}$	$54.87_{\pm 0.33}$	$44.88_{\pm 0.24}$	$0.00_{\pm 0.00}$	$70.64_{\pm 0.38}$	$56.16_{\pm 0.27}$
Non-neg.	NCL-ReLU	×	×	$95.44_{\pm 0.06}$	$52.90_{\pm 0.07}$	$46.55_{\pm 0.12}$	$94.72_{\pm 0.10}$	$75.90_{\pm 0.52}$	$62.84_{\pm 0.43}$
	NCL-RepReLU	×	×	$97.15_{\pm 0.17}$	$53.88_{\pm 0.47}$	$42.33_{\pm 0.44}$	$95.21_{\pm 0.13}$	$75.97_{\pm 0.18}$	$60.90_{\pm 0.15}$
Rect. LpJEPA	$\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$	×	×	$44.33_{\pm 0.09}$	$57.24_{\pm 0.78}$	$48.13_{\pm 0.57}$	$26.14_{\pm 0.28}$	$76.88_{\pm 0.26}$	$69.86_{\pm 0.69}$
	$\mathcal{RGN}_1(-2.0, \sigma_{\mathrm{GN}})$	×	×	$94.31_{\pm 0.01}$	$49.14_{\pm 0.54}$	$37.31_{\pm 0.20}$	$93.51_{\pm 0.08}$	$72.53_{\pm 1.41}$	$63.25_{\pm 0.54}$
	$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	×	×	$95.49_{\pm 0.04}$	$44.56_{\pm 0.76}$	$32.28_{\pm 0.30}$	$95.62_{\pm 0.20}$	$70.98_{\pm 0.17}$	$61.80_{\pm 0.10}$
Top-k	Fixed (k=20/80)	×	×	$96.09_{\pm 0.00}$	$53.93_{\pm 0.34}$	$37.93_{\pm 0.16}$	$96.09_{\pm 0.00}$	$74.86_{\pm 0.37}$	$62.62_{\pm 0.15}$
	Annealed (k=20/80)	×	×	$96.09_{\pm 0.00}$	$56.16_{\pm 0.52}$	$41.40_{\pm 0.32}$	$96.09_{\pm 0.00}$	$75.66_{\pm 0.56}$	$64.58_{\pm 0.39}$
S^2-JEPA (ours)	Probe-guided	✓	✓	$96.17_{\pm 0.03}$	$60.75_{\pm 0.47}$	$48.83_{\pm 0.61}$	$94.54_{\pm 0.04}$	$75.94_{\pm 0.14}$	$61.36_{\pm 0.31}$
	Label-free	×	✓	$96.02_{\pm 0.04}$	$60.62_{\pm 0.29}$	$48.38_{\pm 0.17}$	$95.07_{\pm 0.03}$	$75.22_{\pm 0.17}$	$60.36_{\pm 0.13}$
	Reference-free	×	×	$95.91_{\pm 0.02}$	$58.97_{\pm 0.31}$	$46.73_{\pm 0.35}$	$95.26_{\pm 0.39}$	$74.73_{\pm 0.86}$	$66.04_{\pm 0.38}$

Levels of autonomy. We evaluate S^2-JEPA with three levels of autonomy: probe-guided, label-free, and reference-free. The probe-guided variant uses a labeled probe for sparsity pacing and a dense reference for LR scaling. The label-free variant replaces the labeled probe with loss curvature. The default reference-free variant also replaces the dense reference with warm-up density, requiring neither labels nor an additional run. App. A.2 gives the rules for the first two variants.

5 Experiments

We first evaluate whether S^2-JEPA recovers the accuracy lost under fixed sparse targets (§ 5.1) and whether these gains extend across datasets, backbones, and optimizers (§ 5.2). We then isolate the contribution of each component (§ 5.3) and evaluate downstream performance (§ 5.4).

Setup. Following Kuang et al. (2026), we train ResNet-18 (He et al., 2016) on CIFAR-100 (Krizhevsky, 2009) and STL-10 (Coates et al., 2011), ResNet-50 on ImageNet-100 (Deng et al., 2009), and ViT-S (Dosovitskiy et al., 2021) on CIFAR-100. Unless stated otherwise, training lasts 100 epochs, and we report the mean and sample standard deviation over three runs. We evaluate frozen encoder and projector representations with linear probes. Probe hyperparameters are selected on a stratified 10% subset of the training set, then refit on the full training set and evaluated on the official test or validation split. Sparsity is the percentage of exact zeros in the final projector output. Unlike § 3, all results here use frozen evaluation.

Baselines. Dense and fixed-target Rectified LpJEPA share the objective and training recipe of S^2-JEPA but use fixed values of μ and the LR, isolating the effect of our controllers. The top-k baselines retain the k largest coordinates of a Rectified LpJEPA projector (Kuang et al., 2026), either throughout training or by annealing from $k = D$. We also compare with dense LeJEPA, SimCLR (Chen et al., 2020), VICReg (Bardes et al., 2022), NCL-ReLU, and NCL-RepReLU (Wang et al., 2024). We report all three S^2-JEPA autonomy levels from § 4.

5.1 Performance at high sparsity

Table 1 compares the methods at similar sparsity. Fixed-target Rectified LpJEPA loses encoder and projector accuracy near 95% sparsity. The larger drop for $p = 2$ (App. D.1) indicates that performance depends on the target family, not only final sparsity. S^2-JEPA reduces this loss for both targets. On CIFAR-100, its $p = 1$ encoder is 1.7 points above the dense reference, while its $p = 2$ encoder gains 24.2 points over the matched fixed target (App. D.1). On ImageNet-100, it retains

97.2% of dense encoder accuracy and gives the highest projector accuracy among sparse methods. Annealed top-k improves over fixed top-k, supporting delayed sparsification, but remains below S^2-JEPA on CIFAR-100 and in ImageNet-100 projector accuracy. Scheduling alone is therefore insufficient. NCL is competitive in ImageNet-100 encoder accuracy but weaker in projector accuracy and on CIFAR-100. Dense SimCLR and VICReg have stronger projectors, and VICReg has the strongest encoder, although these methods use different objectives. On ImageNet-100, S^2-JEPA (95.3%) beats fixed targets on both sides of its sparsity: 93.5% ($\mu = -2$) and 95.6% ($\mu = -2.5$).

The autonomy variants clarify the value of each external signal. Curvature remains within 0.7 encoder points of labelled-probe pacing on both datasets, so labels provide little benefit for controlling sparsity. The dense reference improves encoder accuracy by 1.7 points on CIFAR-100 but reduces projector accuracy by 4.7–5.7 points on ImageNet-100. Its benefit therefore does not transfer consistently. This agrees with the always-on ablation, where continued LR correction lowers ImageNet-100 projector accuracy (§ 5.3). The reference-free variant disables correction once movement shortfalls disappear, providing a default without labels or reference runs.

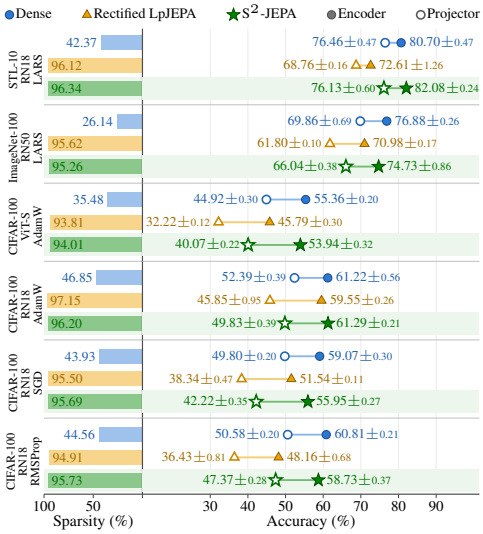

Figure 6: **Generalization across training settings.** Evaluation sparsity (left, %) and frozen encoder (filled) and projector (hollow) accuracy (right, %) after 100 epochs. Each block changes the dataset, backbone, or optimizer and uses its own dense ($\mu = 0$) and fixed-target (Rectified LpJEPA, $\mu = -2.5$, $p = 1$) baselines. S^2-JEPA is reference-free. Magnitude correction remains active only for LARS on CIFAR-100 and STL-10. In all other settings, the checks keep $\alpha = 1$ (from epoch 18 on ImageNet-100), showing that the gains come from sparsity pacing alone.

5.2 Generalization

Fig. 6 varies the dataset, backbone, and optimizer without retuning S^2-JEPA. The fixed-target encoder gap ranges from 1.7 points with AdamW to 12.7 with RMSProp, showing that corrections do not transfer uniformly across settings. S^2-JEPA improves over the fixed target in all seven settings and recovers 59–117% of the gap. Outside LARS, the first check keeps $\alpha = 1$, so these gains come from sparsity pacing alone. Recovery is lowest with SGD because the first eligibility check keeps $\alpha = 1$, although SGD does not rescale smaller gradients.

Projector accuracy is harder to recover. After 100 epochs, it remains below dense accuracy in every setting, with gaps larger than those of the encoder except for a 0.3-point gap on STL-10. Longer training shows that this loss also comes from optimization (App. D.4). At 1000 CIFAR-100 epochs, the fixed-target encoder matches dense accuracy (65.1% vs. 65.3% at 97.4% sparsity), but its projector is 4.8 points lower; label-free S^2-JEPA closes this gap (61.5% vs. 61.3% at 95.9% sparsity). On ImageNet-100, reference-free S^2-JEPA is within 0.5 points of dense training with 20× fewer active coordinates.

5.3 Ablations

Table 2 shows that the two controllers are complementary. Sparsity pacing alone recovers 9.4 encoder points on CIFAR-100 and 5.7 on STL-10, while magnitude correction adds a further 5.0 and 3.8 points after pacing reduces density. With a fixed target, the warm-up is already sparse, so the density ratio provides little correction and the second check disables it at epoch 18 in all six runs.

The eligibility checks prevent harmful corrections. On ImageNet-100, the second check disables magnitude correction, keeping the full method close to pacing alone. Forcing it to remain active leaves encoder accuracy nearly unchanged but lowers projector accuracy by 3.1 points and ViT-S encoder accuracy by 2.3 points (App. D.3). Thus, a correction that gains 5.0 points on CIFAR-100 can hurt in other settings, which the checks detect from the current run. With its best LR multiplier, the fixed-target baseline comes within 0.7 encoder points of S^2-JEPA but remains 2.1 projector points lower, while each candidate requires a full training run (App. D.2).

Table 2: **Contribution of sparsity pacing and magnitude correction.** Frozen encoder and projector accuracy and evaluation sparsity (%) after 100 epochs with LARS and $p = 1$. Without sparsity pacing, the target mean is fixed at $\mu = -2.5$ from the first update. Standard magnitude correction uses both eligibility checks, while "always on" removes them. On CIFAR-100 and STL-10, both checks retain the correction, so the always-on setting produces the same run as the full method.

S²-JEPA components		CIFAR-100 / ResNet-18			STL-10 / ResNet-18			ImageNet-100 / ResNet-50		
Sparsity pacing	Magnitude correction	Sparsity (%) ↑	Encoder ↑	Projector ↑	Sparsity (%) ↑	Encoder ↑	Projector ↑	Sparsity (%) ↑	Encoder ↑	Projector ↑
×	×	$95.49_{\pm 0.04}$	$44.56_{\pm 0.76}$	$32.28_{\pm 0.30}$	$96.12_{\pm 0.03}$	$72.61_{\pm 1.26}$	$68.76_{\pm 0.16}$	$95.62_{\pm 0.20}$	$70.98_{\pm 0.17}$	$61.80_{\pm 0.10}$
×	✓	$95.56_{\pm 0.06}$	$44.89_{\pm 1.06}$	$32.60_{\pm 0.69}$	$96.13_{\pm 0.06}$	$73.78_{\pm 0.16}$	$69.03_{\pm 0.53}$	$95.46_{\pm 0.15}$	$71.12_{\pm 0.41}$	$62.31_{\pm 0.35}$
✓	×	$95.95_{\pm 0.04}$	$53.93_{\pm 0.89}$	$40.58_{\pm 0.41}$	$96.34_{\pm 0.06}$	$78.28_{\pm 0.61}$	$73.34_{\pm 0.16}$	$95.10_{\pm 0.04}$	$75.09_{\pm 0.50}$	$65.79_{\pm 0.68}$
✓	always on	*same run as full*			*same run as full*			$95.08_{\pm 0.06}$	$74.95_{\pm 0.39}$	$62.90_{\pm 0.68}$
✓	✓	$95.91_{\pm 0.02}$	$58.97_{\pm 0.31}$	$46.73_{\pm 0.35}$	$96.34_{\pm 0.01}$	$82.08_{\pm 0.24}$	$76.13_{\pm 0.60}$	$95.26_{\pm 0.39}$	$74.73_{\pm 0.86}$	$66.04_{\pm 0.38}$

Extended ablations further examine the behavior and robustness of these components. We study probe-based pacing, ramp scaling, target switching and reversal, and sensitivity to the pacing constants (App. D.2). We also test magnitude correction in five settings and manually rescale the AdamW learning rate (App. D.3). Additional experiments vary the target mean for $p = 1$ and $p = 2$ (App. D.1), extend training to 1000 epochs (App. D.4), evaluate transfer and sparsity on five new datasets (App. D.5), and compare retrieval and dependence between coordinates (App. D.6).

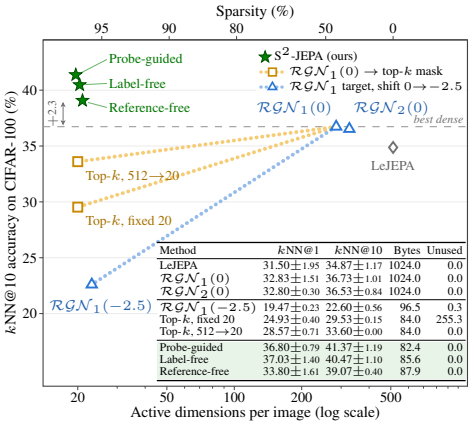

Figure 7: **Retrieval accuracy vs. active dimensions on CIFAR-100.** Dotted lines join the dense Rectified LpJEPA model to its sparse versions; Inset: kNN mean±s.d. over three runs, bytes per image (fp16 values, uint16 indices, int32 row pointers), and coordinates never active on the test set.

5.4 Downstream Performance

We directly evaluate the projector codes using nearest-neighbor retrieval (Fig. 7). All S²-JEPA variants outperform the dense JEPA baselines using only about 20 of 512 coordinates per image. The reference-free variant improves kNN@10 by 2.3 points over the strongest dense model with 12× less storage, and matches dense accuracy on STL-10 with 13× less storage (App. D.6). Fixed top-k leaves about half of its coordinates unused, while annealed top-k and S²-JEPA use the full set, supporting the capacity effect in § 3.2. At similar code sizes, PCA loses about 4.7 kNN@10 points, while int8 quantization retains accuracy but uses about 6× more storage (Tab. A14).

Limitations. The first check relies on layerwise update scaling, which LARS exposes. Extending it to SGD is not straightforward, as enabling magnitude correction improves frozen encoder accuracy by 5.7 points but reduces online accuracy by 1.8 points (App. D.3). A parameter-movement criterion could generalize the check across optimizers, but handling such conflicting outcomes remains open. More broadly, our experiments cover only up to ImageNet-100 with ResNet-50, so the validity of the same constants at larger scales is left for future work.

6 Conclusion

We show that the sparsity cliff in JEPA is largely an optimization cost rather than a limit of sparse representations. Sparse targets reduce parameter movement, while early sparsification restricts feature formation. S²-JEPA addresses both by adapting sparsity and update magnitude using signals from the current run, without labels or reference runs. Across various configurations, it improves over fixed-sparse Rectified LpJEPA by up to 14.4 encoder points. At about 96% sparsity, it matches or exceeds dense encoder accuracy and transfer on CIFAR-100 and STL-10, while reducing retrieval storage by 12×. Our results show sparse codes can retain (and also in certain cases, improve) performance while reducing the storage needed for large embedding collections.

AI USE STATEMENT

We used generative AI tools to polish language and improve clarity in the writing of this manuscript, including refining phrasing, tightening prose, and improving readability of technical descriptions. We did not use generative AI tools to generate research ideas, design experiments, produce results, or write code; all methodology, experiments, and analyses are the authors' own. All AI-assisted text was reviewed and edited by the authors for accuracy and to ensure it faithfully represents our work. We take full responsibility for the final content of this paper, including all text, claims, and artifacts.

ETHICS STATEMENT

This work studies self-supervised visual representation learning and sparse embeddings using standard image classification benchmarks. We do not collect new data or use personally identifiable or sensitive information beyond what may already be present in the original datasets. Our method is intended to improve the efficiency of representation learning and reduce the storage cost of large embedding collections. We do not identify any direct ethical risks specific to this work beyond those associated with the datasets and downstream use of learned visual representations.

REPRODUCIBILITY STATEMENT

§ 4 and Algorithm 1 define S^2-JEPA, while App. A.1 lists all controller constants and explains how curvature and eligibility are measured. App. A.2 gives the rules for the probe-guided and label-free variants. App. C and Tab. A3 provide the training setup, hyperparameters, and evaluation, retrieval, and storage protocols. We report means and sample standard deviations over three seeds. We will release the code and configurations for all experiments upon publication.

REFERENCES

Mohammed Adnan, Rohan Jain, Tom Jacobs, Ekansh Sharma, Rahul G Krishnan, Rebekka Burkholz, and Yani Ioannou. Sparseopt: Addressing normalization-induced gradient skew in sparse training. In *Forty-third International Conference on Machine Learning*, 2026.

Ahmad Aloradi, Tim Roith, Emanuël A. P. Habets, and Daniel Tenbrinck. Adaptive regularization for sparsity control in bregman-based optimizers, 2026.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 2025.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Nolan Dey, Shane Bergsma, and Joel Hestness. Sparse maximal update parameterization: A holistic approach to sparse training dynamics. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 33836–33862. Curran Associates, Inc., 2024. doi: 10.52202/079017-1066.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2021.

Utku Evci, Yani A. Ioannou, Cem Keskin, and Yann Dauphin. Gradient flow in sparse neural networks and how lottery tickets win, 2022.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

Lixuan Guo, Yifei Wang, Tiansheng Wen, Yifan Wang, Aosong Feng, Bo Chen, Stefanie Jegelka, and Chenyu You. CSRv2: Unlocking ultra-sparse embeddings. In *The Fourteenth International Conference on Learning Representations*, 2026.

Max Hartman and Lav Varshney. Sparsejepa: Sparse representation learning of joint embedding predictive architectures, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024.

András Kalapos and Bálint Gyires-Tóth. Cnn-jepa: Self-supervised pretraining convolutional neural networks using joint embedding predictive architecture. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pp. 1111–1114, 2024. doi: 10.1109/ICMLA61862.2024.00169.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yilun Kuang, Yash Dagade, Tim G. J. Rudner, Randall Balestriero, and Yann LeCun. Rectified lpJEPA: Joint-embedding predictive architectures with sparse and maximum-entropy representations. In *Forty-third International Conference on Machine Learning*, 2026.

Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5533–5543. PMLR, 13–18 Jul 2020.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity, 2020.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, December 2022.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Shiyu Liu, Rohan Ghosh, John Chong Min Tan, and Mehul Motani. Optimizing learning rate schedules for iterative pruning of deep neural networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Alireza Makhzani and Brendan Frey. k-sparse autoencoders, 2014.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. ReLU strikes back: Exploiting activation sparsity in large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. AC/DC: Alternating compressed/decompressed training of deep neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020.

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18098–18109. Curran Associates, Inc., 2020.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, and Maosong Sun. ProSparse: Introducing and enhancing intrinsic activation sparsity within large language models. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 2626–2644, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

Yifei Wang, Qi Zhang, Yaoyu Guo, and Yisen Wang. Non-negative contrastive learning, 2024.

Tiansheng Wen, Yifei Wang, Zequn Zeng, Zhong Peng, Yudi Su, Xinyang Liu, Bo Chen, Hongwei Liu, Stefanie Jegelka, and Chenyu You. Beyond matryoshka: Revisiting sparse coding for adaptive representation. In *Forty-second International Conference on Machine Learning*, 2025.

Jheng-Hong Yang, Xueguang Ma, and Jimmy Lin. Sparsifying sparse representations for passage retrieval by top-k masking, 2021.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Michael H. Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression, 2018.

APPENDIX

App. A provides the controller constants, the probe-guided and label-free rules, and the proof of Proposition 1. App. B discusses additional related work. App. C describes the training and evaluation protocols. App. D contains the additional results cited in the main text.

A METHOD DETAILS

A.1 CONTROLLER CONSTANTS

Table A1 lists all constants used by S^2-JEPA. We use the same values across datasets, backbones, and optimizers. The floor $\mu_{\min}$ limits how far the target mean can decrease. We use $\mu_{\min} = -2.5$ on CIFAR-100 and STL-10, including the ViT-S experiments. On ImageNet-100, $\mu = -2.5$ leaves a larger fraction of coordinates active, so we use $\mu_{\min} = -3$. For 1000-epoch runs, the target is updated every ten epochs rather than every epoch. All other constants, including the ten-epoch warm-up, remain unchanged.

Table A1: **S^2-JEPA constants.** One set of values is used in every setting; the last column lists the only changes, which follow the training budget.

Component	Constant	Value	1000 epochs
Warm-up	length (epochs)	10	10
Sparsity pacing	update interval (epochs)	1	10
	step sizes δ	0.025 / 0.05	–
	slope window / fast-step threshold	5 / -0.02	–
	spike multiplier / noise window	2 / 10	–
	floor $\mu_{\min}$	-2.5 (IN-100: -3)	–
Magnitude correction	damping β	0.7	–
	cap $\alpha_{\max}$	10	–
	per-epoch change in α / floor	$\times[0.5, 2]$ / 0.1	–
	density EMA window	5	–
	ramp step / cap	$\times 1.05$ / 6	–
	ramp: curvature-slope window / max	5 / 0	–
	ramp: minimum movement delivery	0.9	–
Eligibility	saturated share of updates (warm-up), min	0.95	–
	second check: epoch	18	–
	mean movement delivery, epochs 11–14, max	0.948	–
	slope of log movement, epochs 8–18, max	-0.005	–

Curvature estimation. We estimate $\lambda_{\max}$ once per epoch using five steps of Hessian-vector power iteration over all encoder and projector parameters. We use a fixed batch of 64 training images and a fixed target draw so that the same function is measured each epoch. The estimation runs in fp32 on a detached copy of the model and does not affect training. If the estimate is not finite, the controller holds μ and α for that epoch. If it is non-positive, the log-slope is undefined, so the controller takes the smaller step and holds the ramp. Curvature estimation adds no measurable time per epoch, while all controller measurements together add about 11% overhead on CIFAR-100 and 17% on ImageNet-100.

Eligibility quantities. The saturated share is the fraction of LARS-scaled tensors whose trust ratio reaches the layerwise cap, averaged over the warm-up. Optimizers without layerwise scaling have no such quantity, so they fail the first check. Delivery is the parameter movement achieved in an epoch divided by its target. The second check averages this quantity over epochs 11–14, the first epochs for which a target is available. The eligibility thresholds were set using archived CIFAR-100 and ImageNet-100 traces, checked on STL-10, and frozen before any reported run. ViT-S, SGD, and RMSProp were not used to set them. The resulting decisions are also robust to the exact thresholds (Tab. A2). With the other threshold fixed, all decisions remain unchanged for any delivery threshold of

at least 0.914. The same holds for slope thresholds between -0.016 and $+0.002$, since the movement slope is below -0.016 on CIFAR-100 and STL-10 and above -0.004 on ImageNet-100.

Table A2: **Second eligibility check across runs.** Mean movement delivery over epochs 11–14 and slope of log movement over epochs 8–18 for all reference-free S^2-JEPA runs with LARS (ranges over three seeds). The check keeps magnitude correction only if delivery is at most 0.948 and the slope is at most -0.005.

Setting	Mean delivery	Slope of log movement	Decision
CIFAR-100 / ResNet-18	0.909–0.910	-0.027 to -0.021	keep (3/3)
STL-10 / ResNet-18	0.913	-0.017	keep (3/3)
ImageNet-100 / ResNet-50	0.944–0.954	-0.003 to $+0.005$	stop (3/3)

A.2 Probe-guided and label-free rules

Probe-guided pacing. The probe-guided variant replaces curvature with linear-probe accuracy, measured once per epoch. On CIFAR-100, the probe uses 50 labelled images per class that are excluded from representation training. After a ten-epoch warm-up at $\mu = 0$, the controller lowers μ by 0.05 while accuracy improves and by 0.025 when improvement slows. Unlike curvature pacing, it also increases μ if accuracy falls below its running best.

Dense-reference magnitude correction. The label-free and probe-guided variants calibrate the LR using a separate dense run. Let ΔP_t denote the encoder and projector update path during epoch t. Given a target $\Delta P_t^\star$, the controller sets

$$\alpha_{t+1} = \min\left\{\alpha_{\max},\, \alpha_t \left(\frac{\Delta P_t^\star}{\Delta P_t}\right)^\beta\right\}, \qquad \beta = 0.7, \quad \alpha_{\max} = 10, \tag{8}$$

with $\Delta P_t^\star = \kappa \Delta P_t^{\text{dense}}$ and $\kappa = 2.5$ on CIFAR-100 (3 for 1000-epoch runs). Each new training setting requires its own dense run.

Relation to the reference-free rule. Define the rescaled movement as $\widehat{\Delta P}_t = \Delta P_t / \alpha_t$. Substituting $\Delta P_t^\star = r_t m_t^\star \widehat{\Delta P}_t$ into Equation (8) gives Equation (7). The observed path cancels, leaving a smoothed correction based on representation density rather than a target from a separate run. This $\Delta P_t^\star$ is the movement target used in § 4.2. Its delivery ratio is the movement achieved in the next epoch divided by this target. Both the ramp and the second eligibility check use this ratio.

A.3 Proof of Proposition 1

Proof of Proposition 1. Let $\nu_\mu = \mathcal{GN}_p(\mu, \sigma)$, so that ρ_μ is its rectification. Coupling samples of ν_μ and $\nu_{\mu'}$ through the same location-family noise displaces each sample by exactly Δ, so $W_2(\nu_\mu, \nu_{\mu'}) \le \Delta$; since the difference in means is a lower bound on W_2, this holds with equality. Rectification is 1-Lipschitz, so the same coupling gives $W_2(\rho_\mu, \rho_{\mu'}) \le \Delta$. Over D independent coordinates, each coordinate moves by at most Δ, so the Euclidean displacement is at most $\sqrt{D}\Delta$, and the projected displacement along a unit vector c is at most $\sum_i |c_i|\Delta = \|c\|_1 \Delta$. □

B Additional related work

This section places S^2-JEPA within two lines of work: sparse representation learning and optimization and scheduling under sparsity. Much of the existing work focuses on weight sparsity or codes added after pretraining (Hoefler et al., 2021). In contrast, we study activation sparsity learned during self-supervised pretraining.

B.1 Sparse representation learning

Sparse representations have a long history, from sparse coding of natural images (Olshausen & Field, 1996) to rectifier networks (Glorot et al., 2011). Activation sparsity is also used to improve inference

efficiency in CNNs and LLMs, often through regularization or modified activation functions (Kurtz et al., 2020; Mirzadeh et al., 2024; Song et al., 2025). Sparse autoencoders and top-k codes (Makhzani & Frey, 2014; Huben et al., 2024; Gao et al., 2025), as well as compact embeddings (Kusupati et al., 2022; Wen et al., 2025), learn sparse representations for interpretability or retrieval. These methods provide direct control over the number of active units, but typically start from a densely pretrained model and introduce sparsity afterwards.

Closer to our setting are methods that learn sparse or non-negative representations during self-supervised training. Non-negative contrastive learning (Wang et al., 2024) constrains contrastive features to be non-negative, which induces sparsity. SparseJEPA (Hartman & Varshney, 2025) adds a penalty that encourages latent variables to be shared across semantically related features. Rectified LpJEPA (Kuang et al., 2026) matches features to a rectified generalized Gaussian whose target distribution determines the expected sparsity. We build on this formulation because its location μ provides direct control over sparsity. For post-hoc codes, CSRv2 (Guo et al., 2026) reports dead neurons and unstable training at very small k, which it addresses through k-annealing and a supervised contrastive loss. S^2-JEPA instead studies why sparse JEPA representations learned directly during pretraining degrade at extreme sparsity, and adapts the training process rather than introducing a new sparsity objective.

B.2 Optimization and scheduling under sparsity

Sparsity also changes optimization. Sparse networks can suffer from poor gradient flow at initialization (Evci et al., 2022), normalization can skew gradients across neurons (Adnan et al., 2026), and the best learning rate and initialization can vary with density (Dey et al., 2024). Learning rate schedules after pruning also affect performance (Renda et al., 2020; Liu et al., 2023). These methods address such effects using rules defined in advance from density or a fixed schedule, primarily for weight sparsity. Our square-root factor is a similar density-based correction. S^2-JEPA adds feedback to determine whether this correction is needed—it applies correction only when the optimizer's layerwise scaling is saturated, keeps it while the run misses its movement targets and movement continues to fall, and increases it only while training remains stable (§ 4.2).

Several methods also introduce sparsity gradually, including gradual magnitude pruning (Zhu & Gupta, 2018), pruning with regrowth (Liu et al., 2021), progressive sparsity regularization in LLMs (Song et al., 2025), and k-annealing for ultra-sparse embeddings (Guo et al., 2026). AC/DC (Peste et al., 2021) instead alternates between dense and sparse phases. Our results similarly show that delaying sparsification helps; a linear schedule alone raises CIFAR-100 encoder accuracy from 44.56 to 53.47 (Tab. A6), while abrupt switches between dense and sparse targets perform worse than gradual schedules in our setting (Tab. A8). Prior schedules map training time to a sparsity level chosen in advance, requiring the pace to be tuned for each setting. S^2-JEPA instead uses training dynamics to choose the next target, taking larger steps while curvature falls, smaller steps otherwise, and holding μ after a sharp change.

Other methods adapt sparsity using feedback. Dynamic pruning with feedback (Lin et al., 2020) uses updates from dense weights to reactivate pruned weights, while RigL (Evci et al., 2021) uses gradients to select connections for regrowth. Curvature-based pruning (LeCun et al., 1989; Singh & Alistarh, 2020) uses second-order information to decide which weights to remove. STR (Kusupati et al., 2020) learns layerwise thresholds through gradient descent, while Aloradi et al. (2026) adjust the ℓ_1 strength so that measured sparsity tracks a target. In these methods, feedback determines which units are zero or how a sparsity level is reached. In S^2-JEPA, the final target is fixed by $\mu_{\min}$; feedback instead controls how quickly it is approached and how much the update is corrected. S^2-JEPA therefore uses two feedback loops: curvature controls target pacing, while optimizer saturation and parameter movement gate the LR correction.

C Experimental setup

Training. All runs follow the Rectified LpJEPA recipe (Kuang et al., 2026) and use two augmented views per image. The projector dimension is $D = 512$, or $D = 2048$ on ImageNet-100, and the target uses $p = 1$ unless stated otherwise. Each run uses one NVIDIA H200 GPU, 16-bit mixed

precision, and weight decay 10^{-4}. The learning-rate schedule uses a ten-epoch linear warm-up from 3×10^{-5}, followed by cosine decay to zero.

LARS uses a trust coefficient of 0.02, momentum of 0.9, and the layerwise LR cap in Equation (4); biases and normalization parameters are excluded from layerwise scaling. AdamW uses moment coefficients of 0.9 and 0.999. SGD and RMSProp use momentum of 0.9, and RMSProp uses a smoothing constant of 0.99. Table A3 lists the settings that vary across runs.

The augmentations comprise random resized crops with scale 0.2–1, color jitter with parameters $(0.4, 0.4, 0.2, 0.1)$ applied with probability 0.8, grayscale with probability 0.2, solarization with probability 0.1, and horizontal flips with probability 0.5. ImageNet-100 additionally uses Gaussian blur with probability 0.5, while the ViT-S runs omit solarization.

Table A3: **Per-setting training hyperparameters.** The online-probe LR is used only for the online accuracy reported in § 3 and in online columns.

Setting	Crop	Batch	Base LR	Online-probe LR
CIFAR-100 / ResNet-18 / LARS	32	256	0.03	0.01
CIFAR-100 / ResNet-18 / AdamW	32	256	0.01	0.005
CIFAR-100 / ResNet-18 / SGD	32	256	0.06	0.01
CIFAR-100 / ResNet-18 / RMSProp	32	256	10^{-4}	0.005
CIFAR-100 / ViT-S / AdamW	32	128	2.5×10^{-4}	0.0025
STL-10 / ResNet-18 / LARS	96	256	0.03	0.01
ImageNet-100 / ResNet-50 / LARS	224	128	0.0825	0.0275

Diagnostics. The diagnostic experiments in § 3 use CIFAR-100, ResNet-18, 100 training epochs, and online encoder top-1 accuracy. All other experiments use frozen evaluation.

Evaluation. We train frozen linear probes on encoder features and projector outputs. The probe learning rate and training duration are selected using a stratified 10% split of the training set. We then refit the probe on the full training set and evaluate it on the official test or validation split. Sparsity is the percentage of exact zeros in the projector output, measured on the evaluation split using the final checkpoint in evaluation mode.

Nearest-neighbour retrieval uses cosine similarity between ℓ_2-normalized projector outputs. kNN@1 assigns the label of the nearest gallery image, while kNN@10 uses a majority vote among the ten nearest images. Ties are assigned to the lower-indexed class. On CIFAR-100, a fixed random permutation partitions the $10{,}000$ test images into $9{,}000$ gallery images and $1{,}000$ queries. Storage for sparse outputs includes fp16 values, uint16 indices, and one int32 row pointer per image. Storage for dense outputs includes only fp16 values.

D Additional results

D.1 Target distributions

Table A4 extends the target comparison in Tab. 1. Among fixed targets, encoder accuracy is highest near $\mu = 0.5$ and decreases as μ becomes smaller, with a sharper decline below $\mu = -1.5$. At matched sparsity, $p = 2$ targets have lower accuracy than $p = 1$ targets. At approximately 96% sparsity, all three S^2-JEPA variants have higher encoder accuracy than every dense target.

Table A5 reports the corresponding projector-code results. For fixed targets, nearest-neighbour accuracy decreases from 33.7% at $\mu = 1$ to 19.5% at $\mu = -2.5$. S^2-JEPA obtains 33.8%–37.0% at similar sparsity. Fixed-target purity is highest at $\mu = -2$ and decreases beyond this point, while S^2-JEPA gives the highest purity. Fixed $p = 2$ targets also leave up to 19 coordinates unused, whereas S^2-JEPA uses every coordinate.

Table A4: **Extended target sweep on CIFAR-100.** Frozen encoder and projector accuracy and evaluation sparsity (%) after 100 epochs with ResNet-18. Shaded rows are dense targets ($\mu \geq 0$). Means and sample standard deviations over three runs.

Target	Sparsity (%) ↑	Encoder ↑	Projector ↑
Rectified LpJEPA, $p = 1$			
$\mathcal{RGN}_1(1.0, \sigma_{\mathrm{GN}})$	10.18 ± 0.09	57.83 ± 0.30	46.72 ± 0.32
$\mathcal{RGN}_1(0.5, \sigma_{\mathrm{GN}})$	22.41 ± 0.24	58.19 ± 0.50	48.35 ± 0.49
$\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$	44.33 ± 0.09	57.24 ± 0.78	48.13 ± 0.57
$\mathcal{RGN}_1(-0.5, \sigma_{\mathrm{GN}})$	66.07 ± 0.13	56.01 ± 0.04	47.24 ± 0.20
$\mathcal{RGN}_1(-1.0, \sigma_{\mathrm{GN}})$	84.30 ± 0.04	54.17 ± 0.23	43.95 ± 0.68
$\mathcal{RGN}_1(-1.5, \sigma_{\mathrm{GN}})$	91.54 ± 0.01	52.68 ± 0.45	41.21 ± 0.33
$\mathcal{RGN}_1(-2.0, \sigma_{\mathrm{GN}})$	94.31 ± 0.01	49.14 ± 0.54	37.31 ± 0.20
$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	95.49 ± 0.04	44.56 ± 0.76	32.28 ± 0.30
$\mathcal{RGN}_1(-3.0, \sigma_{\mathrm{GN}})$	96.30 ± 0.02	35.31 ± 0.61	23.79 ± 0.70
Rectified LpJEPA, $p = 2$			
$\mathcal{RGN}_2(0, \sigma_{\mathrm{GN}})$	36.21 ± 0.09	57.58 ± 0.01	48.16 ± 0.27
$\mathcal{RGN}_2(-2.0, \sigma_{\mathrm{GN}})$	94.25 ± 0.08	38.27 ± 0.15	27.59 ± 0.51
$\mathcal{RGN}_2(-2.25, \sigma_{\mathrm{GN}})$	95.47 ± 0.01	31.80 ± 1.09	22.56 ± 0.32
$\mathcal{RGN}_2(-2.5, \sigma_{\mathrm{GN}})$	96.14 ± 0.15	27.19 ± 0.17	18.76 ± 0.17
S^2-JEPA (ours)			
Probe-guided, $p = 1$	96.17 ± 0.03	60.75 ± 0.47	48.83 ± 0.61
Label-free, $p = 1$	96.02 ± 0.04	60.62 ± 0.29	48.38 ± 0.17
Reference-free, $p = 1$	95.91 ± 0.02	58.97 ± 0.31	46.73 ± 0.35
Reference-free, $p = 2$	95.91 ± 0.03	56.01 ± 0.41	43.15 ± 0.24

D.2 Sparsity pacing and schedule shape

A natural question is whether the closed loop matters, or whether a fixed schedule with a tuned LR would do as well. Table A6 tests this with open-loop recipes tuned on CIFAR-100. On CIFAR-100 they come within 0.4 encoder points of S^2-JEPA, although they end sparser, so a recipe tuned on one setting can match the controller there. The same recipes lose about 3 projector points on ImageNet-100, and the fixed target with its CIFAR multiplier loses 5, because the LR correction keeps running after it stops helping. The eligibility checks detect this from the run and switch the correction off. Without any correction, linear and cosine anneals also fall below curvature pacing on both datasets, so curvature is at least as good as a hand-chosen shape without being tuned.

Table A7 compares the pacing signals and components of magnitude correction. Curvature and the labelled probe reach similar frozen encoder accuracy at similar sparsity. Adding the ramp improves encoder accuracy by 3.5 points over the square-root scale alone. A fixed target with the best constant multiplier from the sweep in Fig. 2 comes within 0.7 encoder points of S^2-JEPA but stays 2.1 projector points below it. Finding that multiplier took a full sweep, and the best value differs between dense and sparse targets (§ 3.1), whereas S^2-JEPA reaches this accuracy in a single run. Table A8 compares gradual pacing with abrupt changes between dense and sparse targets. Longer dense training before the switch improves accuracy, but even a 40-epoch dense phase remains below gradual pacing. Adding a dense phase after sparse training does not recover the lost accuracy, even when the run ends with a dense target. The inverted schedule also ends dense but has lower encoder accuracy than gradual pacing. These results show that both the order and rate of sparsification matter, rather than only the number of dense epochs. Tuning the dense target's LR in the same way barely changes it (57.3% encoder accuracy at its best multiplier), so S^2-JEPA's 1.7-point encoder margin over dense training does not come from an under-tuned dense baseline.

Table A9 shows that halving or doubling the pacing constants changes online encoder accuracy by at most 0.4 points. Allowing μ to increase toward dense leaves the final codes at only 53–56% sparsity instead of 96%. A prescribed exponential schedule reaches the same final sparsity as curvature pacing but gives 4–8 points lower online encoder accuracy, with or without magnitude correction.

Table A5: **Projector representations across target distributions.** CIFAR-100 evaluation sparsity, active coordinates per image, kNN@1 accuracy, purity@64, and coordinates unused across the evaluation set. Accuracy, sparsity, and purity are percentages. Both $p = 1$ and $p = 2$ use the rectified projector output. Purity@k takes the k most frequently active coordinates, assigns each its majority class on a random half of the evaluation images, and reports the fraction of its activations on the other half whose label matches that class; coordinates active on fewer than 1% of the first half are excluded. We use $k = 64$. Bold and underline mark the best and second-best kNN@1 and purity.

Target	Sparsity (%) ↑	Active	kNN@1 ↑	Purity ↑	Unused ↓
Rectified LpJEPA, $p = 1$					
$\mathcal{RGN}_1(1.0, \sigma_{\mathrm{GN}})$	10.18 ± 0.09	459.86 ± 0.45	33.70 ± 0.87	1.05 ± 0.00	0.0
$\mathcal{RGN}_1(0.5, \sigma_{\mathrm{GN}})$	22.41 ± 0.24	397.27 ± 1.23	33.50 ± 0.36	1.16 ± 0.00	0.0
$\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$	44.33 ± 0.09	285.05 ± 0.44	32.83 ± 1.51	1.49 ± 0.01	0.0
$\mathcal{RGN}_1(-0.5, \sigma_{\mathrm{GN}})$	66.07 ± 0.13	173.73 ± 0.68	31.07 ± 1.27	2.08 ± 0.05	0.0
$\mathcal{RGN}_1(-1.0, \sigma_{\mathrm{GN}})$	84.30 ± 0.04	80.41 ± 0.23	29.00 ± 0.46	3.57 ± 0.08	0.0
$\mathcal{RGN}_1(-1.5, \sigma_{\mathrm{GN}})$	91.54 ± 0.01	43.31 ± 0.07	27.13 ± 0.61	5.23 ± 0.09	0.0
$\mathcal{RGN}_1(-2.0, \sigma_{\mathrm{GN}})$	94.31 ± 0.01	29.16 ± 0.04	25.00 ± 0.46	6.37 ± 0.36	0.0
$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	95.49 ± 0.04	23.11 ± 0.19	19.47 ± 0.23	6.25 ± 0.21	0.3
$\mathcal{RGN}_1(-3.0, \sigma_{\mathrm{GN}})$	96.30 ± 0.02	18.95 ± 0.12	14.30 ± 0.87	4.85 ± 0.35	0.7
Rectified LpJEPA, $p = 2$					
$\mathcal{RGN}_2(0, \sigma_{\mathrm{GN}})$	36.21 ± 0.09	326.60 ± 0.44	32.80 ± 0.30	1.34 ± 0.01	0.0
$\mathcal{RGN}_2(-2.0, \sigma_{\mathrm{GN}})$	94.25 ± 0.08	29.45 ± 0.41	15.30 ± 0.75	4.64 ± 0.18	5.3
$\mathcal{RGN}_2(-2.25, \sigma_{\mathrm{GN}})$	95.47 ± 0.01	23.21 ± 0.05	11.37 ± 0.50	4.36 ± 0.14	19.0
$\mathcal{RGN}_2(-2.5, \sigma_{\mathrm{GN}})$	96.14 ± 0.15	19.78 ± 0.75	9.53 ± 0.83	4.18 ± 0.10	9.0
S^2-JEPA (ours)					
Probe-guided	96.17	19.6	$\underline{36.80 \pm 0.79}$	$\mathbf{8.34 \pm 0.44}$	0.0
Label-free	96.02	20.4	$\mathbf{37.03 \pm 1.40}$	$\underline{8.17 \pm 0.14}$	0.0
Reference-free	95.91	21.0	33.80 ± 1.61	8.09 ± 0.20	0.0

Table A6: **Open-loop schedules.** Frozen encoder and projector accuracy and CIFAR-100 evaluation sparsity (%) after 100 epochs; means and sample standard deviations over three runs. Open-loop anneals lower μ from 0 to its floor between epochs 10 and 70, roughly the trajectory S^2-JEPA follows on CIFAR-100. "Best constant" is the LR multiplier selected by a sweep on CIFAR-100 and reused unchanged on ImageNet-100; "always-on correction" is the S^2-JEPA magnitude rule without its eligibility checks.

		CIFAR-100			ImageNet-100	
Schedule	Magnitude	Sparsity	Encoder	Projector	Encoder	Projector
Fixed $\mu = -2.5$	none	$95.49_{\pm 0.04}$	$44.56_{\pm 0.76}$	$32.28_{\pm 0.30}$	$70.98_{\pm 0.17}$	$61.80_{\pm 0.10}$
Fixed $\mu = -2.5$	best constant ($\times 8$)	$97.55_{\pm 0.01}$	$58.24_{\pm 0.45}$	$44.59_{\pm 0.24}$	$72.86_{\pm 0.41}$	$60.91_{\pm 0.29}$
Linear anneal	none	$95.87_{\pm 0.05}$	$53.47_{\pm 0.39}$	$40.21_{\pm 0.28}$	$74.18_{\pm 0.46}$	$64.72_{\pm 0.34}$
Cosine anneal	none	$95.82_{\pm 0.06}$	$52.81_{\pm 0.44}$	$39.76_{\pm 0.31}$	$73.84_{\pm 0.51}$	$64.31_{\pm 0.42}$
Linear anneal	best constant	$97.21_{\pm 0.09}$	$58.61_{\pm 0.35}$	$46.18_{\pm 0.41}$	$74.92_{\pm 0.38}$	$63.08_{\pm 0.36}$
Linear anneal	always-on correction	$96.94_{\pm 0.07}$	$58.73_{\pm 0.38}$	$46.42_{\pm 0.33}$	$74.61_{\pm 0.48}$	$63.17_{\pm 0.40}$
Curvature pacing	none	$95.95_{\pm 0.04}$	$53.93_{\pm 0.89}$	$40.58_{\pm 0.41}$	$75.09_{\pm 0.50}$	$65.79_{\pm 0.68}$
S^2-JEPA, reference-free	with checks	$95.91_{\pm 0.02}$	$58.97_{\pm 0.31}$	$46.73_{\pm 0.35}$	$74.73_{\pm 0.86}$	$66.04_{\pm 0.38}$

Table A7: **Pacing signal and magnitude correction.** Frozen encoder and projector accuracy and evaluation sparsity (%) on CIFAR-100 with ResNet-18 after 100 epochs. The last row is reference-free S^2-JEPA. "Always on" rows apply magnitude correction without the eligibility checks, starting after five warm-up epochs.

Configuration	Sparsity (%) ↑	Encoder ↑	Projector ↑
Pacing signal, with dense-reference magnitude correction			
Labelled probe (probe-guided)	96.17 ± 0.03	60.75 ± 0.47	48.83 ± 0.61
Curvature (label-free)	96.02 ± 0.04	60.62 ± 0.29	48.38 ± 0.17
Fixed target ($\mu = -2.5$), constant LR multiplier, no pacing			
Dense target ($\mu = 0$), $\times 5$ (best in the sweep)	45.90 ± 0.20	57.32 ± 0.25	47.53 ± 0.49
$\times 3.55$ (square-root factor)	96.32 ± 0.05	55.68 ± 0.56	42.74 ± 0.55
$\times 8$ (best in the sweep of Fig. 2)	97.55 ± 0.01	58.24 ± 0.45	44.59 ± 0.24
Reference-free magnitude correction, with curvature pacing			
Square-root factor, always on	95.92 ± 0.01	55.68 ± 0.40	42.48 ± 0.47
+ ramp, always on	95.93 ± 0.02	59.22 ± 0.59	47.25 ± 0.16
+ ramp and eligibility checks	95.91 ± 0.02	58.97 ± 0.31	46.73 ± 0.35

Table A8: **Abrupt changes in the target distribution.** Frozen encoder and projector accuracy and evaluation sparsity (%) on CIFAR-100 after 100 epochs. Dense and sparse phases use $\mu = 0$ and $\mu = -2.5$ with $p = 1$. The first three blocks use no magnitude correction; S^2-JEPA rows are included for reference. Epoch indices start at zero, so the 60–99 dense interval ends training at $\mu = 0$.

Capacity schedule	Sparsity (%) ↑	Encoder ↑	Projector ↑
Initial dense phase, followed by sparsification			
Dense for first 10 epochs	99.03 ± 0.08	40.12 ± 0.37	19.76 ± 0.52
Dense for first 20 epochs	98.25 ± 0.20	47.45 ± 0.82	25.59 ± 1.20
Dense for first 40 epochs	95.09 ± 0.20	52.18 ± 0.42	30.88 ± 0.59
Dense interval inserted after sparse training			
Dense during epochs 60–69	99.19 ± 0.30	41.41 ± 0.76	14.73 ± 1.55
Dense during epochs 60–79	96.25 ± 0.44	46.14 ± 1.18	23.22 ± 1.15
Dense during epochs 60–99	51.82 ± 0.13	49.85 ± 0.50	37.85 ± 0.82
Gradual schedules without magnitude correction			
Sparsity pacing only	95.95 ± 0.04	53.93 ± 0.89	40.58 ± 0.41
Inverted schedule	45.82 ± 0.18	52.56 ± 0.19	44.71 ± 0.96
S^2-JEPA (with magnitude correction)			
Probe-guided	96.17 ± 0.03	60.75 ± 0.47	48.83 ± 0.61
Label-free	96.02 ± 0.04	60.62 ± 0.29	48.38 ± 0.17
Reference-free	95.91 ± 0.02	58.97 ± 0.31	46.73 ± 0.35

Table A9: **Pacing sensitivity and alternative schedules.** Final-epoch online accuracy and training sparsity (%) on CIFAR-100 with ResNet-18 after 100 epochs. Pacing-constant rows use the label-free level; the default row pools six runs and each perturbed row is a single run. Other rows use three runs. Bidirectional pacing lets μ increase again after a curvature spike or a trend reversal.

Configuration	Sparsity (%) ↑	Online enc. ↑	Online proj. ↑
Pacing constants (label-free level)			
Spike multiplier 1	95.94	55.10	45.04
Spike multiplier 2 (default)	95.93 ± 0.07	54.75 ± 0.13	44.99 ± 0.35
Spike multiplier 4	95.84	54.59	44.74
Slope threshold $\times 0.5$	95.83	55.09	44.82
Slope threshold $\times 2$	96.02	54.98	45.55
Bidirectional pacing			
$\mu_{\min} = -3$	53.20 ± 7.14	54.15 ± 0.20	49.12 ± 0.30
$\mu_{\min} = -6$	55.52 ± 6.06	54.31 ± 0.39	49.17 ± 0.36
Prescribed exponential schedule			
Exponential anneal	95.69 ± 0.03	46.87 ± 0.08	35.51 ± 0.45
Exponential anneal, magnitude correction	95.80 ± 0.02	50.87 ± 0.15	39.82 ± 0.37

D.3 Eligibility checks

Table A11 evaluates when magnitude correction should remain active. The checks retain it on CIFAR-100 and STL-10, disable it after epoch 18 on ImageNet-100, and do not activate it for ViT-S with AdamW. On ImageNet-100, all variants use the same ten-epoch warm-up. Keeping correction active has little effect on frozen encoder accuracy but reduces projector accuracy by 3.1 points and online encoder accuracy by 4.3 points. On ViT-S, it reduces frozen encoder accuracy by 2.3 points. Under AdamW, none of the tested constant LR multipliers improves over the uncorrected run (Tab. A10), consistent with the first check setting $\alpha = 1$.

SGD is an exception. Because it has no layerwise update scaling, the first check also sets $\alpha = 1$. However, forcing correction increases frozen encoder accuracy by 5.7 points while reducing online accuracy by 1.8 points. We discuss this limitation in § 5.4.

Table A10: **Learning-rate multipliers under AdamW.** Final-epoch online encoder accuracy (%) for the fixed $\mu = -2.5$ target on CIFAR-100 with ResNet-18, over three runs. A constant multiplier scales the base learning rate throughout training. No multiplier improves on the uncorrected run by more than the run-to-run variation.

LR multiplier	Base LR 10^{-2}	Base LR 3×10^{-3}
$\times 1$ (none)	56.90 ± 0.64	56.73 ± 0.36
$\times 1.5$	56.52 ± 0.48	56.78 ± 0.39
$\times 2$	55.65 ± 0.38	56.95 ± 0.14
$\times 3$	53.51 ± 0.48	56.68 ± 0.28

D.4 Longer training

Table A12 reports results after 1000 epochs, with the target updated every ten epochs. On CIFAR-100, the fixed-target encoder matches its dense reference, but the projector remains 4.8 points lower. Label-free S^2-JEPA also closes the projector gap. On ImageNet-100, reference-free S^2-JEPA is within 0.5 encoder points of the dense reference at 96.2% sparsity. Table A13 shows that the label-free model has higher transfer accuracy than the dense reference on all five downstream datasets, with an average gain of 2.2 points.

D.5 Transfer and sparsity on new domains

We freeze the CIFAR-100 models and fit linear probes on five downstream datasets (Tab. A15). S^2-JEPA improves mean encoder accuracy by 8.7 points over the fixed target and 2.0 points over annealed top-k, while matching its

Table A14: **Retrieval at matched storage on CIFAR-100.** kNN@10 accuracy (%) and bytes per image. PCA is fitted on the gallery images of the dense $\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$ projector and stored in fp16; int8 uses per-vector symmetric quantization with one fp16 scale.

Representation	Bytes ↓	kNN@10 ↑

Table A11: **When to apply magnitude correction.** Frozen and final-epoch online accuracy and evaluation sparsity (%) after 100 epochs. "With eligibility checks" is reference-free S^2-JEPA. On CIFAR-100, STL-10, ViT-S, and SGD, the always-on rows start correction after five warm-up epochs rather than ten. With ten epochs, always-on correction on CIFAR-100 and STL-10 is identical to the full method (Tab. 2), so these rows differ only in warm-up length. On ImageNet-100, all rows use ten warm-up epochs and differ only in correction and eligibility. The checks keep correction active on CIFAR-100 and STL-10, stop it after epoch 18 on ImageNet-100, and keep $\alpha = 1$ for ViT-S with AdamW and for SGD.

Magnitude correction	Sparsity (%) ↑	Frozen enc. ↑	Frozen proj. ↑	Online enc. ↑
CIFAR-100 / ResNet-18 / LARS				
Always on	95.93 ± 0.02	59.22 ± 0.59	47.25 ± 0.16	54.10 ± 0.32
With eligibility checks	95.91 ± 0.02	58.97 ± 0.31	46.73 ± 0.35	53.72 ± 0.41
STL-10 / ResNet-18 / LARS				
Always on	96.33 ± 0.04	82.07 ± 0.63	76.47 ± 0.51	81.86 ± 0.08
With eligibility checks	96.34 ± 0.01	82.08 ± 0.24	76.13 ± 0.60	81.23 ± 0.04
ImageNet-100 / ResNet-50 / LARS				
None	95.10 ± 0.04	75.09 ± 0.50	65.79 ± 0.68	77.71 ± 0.33
Always on	95.08 ± 0.06	74.95 ± 0.39	62.90 ± 0.68	73.44 ± 0.44
With eligibility checks	95.26 ± 0.39	74.73 ± 0.86	66.04 ± 0.38	77.77 ± 0.25
CIFAR-100 / ViT-S / AdamW				
Always on	93.88 ± 0.05	51.62 ± 0.63	39.35 ± 0.42	48.32 ± 0.34
With eligibility checks	94.01 ± 0.03	53.94 ± 0.32	40.07 ± 0.22	51.81 ± 0.29
CIFAR-100 / ResNet-18 / SGD				
Always on	95.65 ± 0.02	61.63 ± 0.42	49.43 ± 0.38	46.35 ± 0.49
With eligibility checks	95.69 ± 0.04	55.95 ± 0.27	42.22 ± 0.35	48.13 ± 0.31

Table A12: **Longer pretraining.** Frozen encoder and projector accuracy and evaluation sparsity (%) after 1000 epochs. The target is updated every ten epochs; all other controller constants are unchanged. The label-free level uses a dense movement reference with $\kappa = 3$.

Configuration	Sparsity (%) ↑	Encoder ↑	Projector ↑
CIFAR-100 / ResNet-18			
Dense reference	35.55 ± 0.32	65.27 ± 0.36	61.30 ± 0.34
$\mathcal{RGN}_1(-1, \sigma_{\mathrm{GN}})$	65.10 ± 0.24	65.73 ± 0.30	61.79 ± 0.25
$\mathcal{RGN}_1(-2, \sigma_{\mathrm{GN}})$	95.13 ± 0.03	65.40 ± 0.40	58.71 ± 0.32
$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	97.39 ± 0.02	65.12 ± 0.09	56.46 ± 0.10
S^2-JEPA, label-free, $\kappa = 3$	95.79 ± 0.04	67.26 ± 0.35	61.49 ± 0.41
ImageNet-100 / ResNet-50			
Dense reference	25.68 ± 0.29	84.58 ± 0.70	78.21 ± 0.58
S^2-JEPA, reference-free	96.24 ± 0.31	84.12 ± 0.14	75.26 ± 0.37

Table A13: **Transfer after longer pretraining.** Frozen encoder accuracy (%) on five downstream datasets after 1000 CIFAR-100 epochs, with projector sparsity measured on CIFAR-100 evaluation images. Mean is the average over the five datasets. Label-free S^2-JEPA uses a dense movement reference with $\kappa = 3$.

Method	Sparsity (%) ↑	DTD ↑	CIFAR-10 ↑	Flowers ↑	Food ↑	Pets ↑	Mean ↑
Dense reference	35.55	35.83 ± 0.27	77.89 ± 0.13	54.01 ± 0.60	26.95 ± 0.41	35.17 ± 0.30	45.97
S^2-JEPA, label-free	95.79	$\mathbf{37.61 \pm 0.95}$	$\mathbf{79.77 \pm 0.16}$	$\mathbf{57.22 \pm 0.52}$	$\mathbf{29.18 \pm 0.18}$	$\mathbf{37.04 \pm 0.40}$	**48.16**

dense reference. After 1000 epochs, the label-free variant is 2.2 points more accurate on average than the dense reference (Tab. A13).

Among models with at least 90% projector sparsity, an S^2-JEPA variant achieves the highest accuracy on each downstream dataset. The reference-free variant also matches or improves on its dense reference across all five datasets, although dense VICReg and SimCLR perform best overall (Tab. A15). Moreover, representations trained at about 96% sparsity on CIFAR-100 remain 95.1–96.0% sparse on each downstream dataset, whereas the sparsity of dense targets varies by up to ten points (Tab. A16). S^2-JEPA therefore retains its storage reduction across domains.

Method	Sparsity (%) ↑	DTD ↑	CIFAR-10 ↑	Flowers ↑	Food ↑	Pets ↑	Mean ↑
Dense references							
SimCLR	0.00	36.47 ± 1.68	78.48 ± 0.48	52.28 ± 1.57	29.47 ± 0.34	37.12 ± 0.43	46.76
VICReg	0.00	37.52 ± 0.57	80.78 ± 0.33	57.02 ± 0.79	30.82 ± 0.16	38.69 ± 0.95	48.97
LeJEPA	0.00	34.93 ± 1.13	76.17 ± 0.24	45.26 ± 2.21	27.15 ± 0.25	31.70 ± 0.86	43.04
$\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$	44.33	36.47 ± 0.87	77.95 ± 0.46	49.61 ± 0.66	28.24 ± 0.08	33.64 ± 0.34	45.18
$\mathcal{RGN}_2(0, \sigma_{\mathrm{GN}})$	36.21	36.13 ± 0.66	77.66 ± 0.38	49.06 ± 1.35	28.10 ± 0.07	32.63 ± 1.34	44.72
Sparse baselines							
NCL-ReLU	95.44	33.58 ± 0.75	75.38 ± 0.24	48.80 ± 0.24	27.42 ± 0.04	32.46 ± 0.31	43.53
NCL-RepReLU	97.15	33.93 ± 0.17	76.17 ± 0.22	49.18 ± 0.45	27.65 ± 0.19	32.59 ± 1.59	43.90
$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	95.49	28.90 ± 1.50	68.74 ± 0.57	41.12 ± 0.81	20.95 ± 0.66	25.11 ± 0.44	36.96
Top-k, fixed 20	96.09	34.45 ± 0.59	76.19 ± 0.76	45.52 ± 0.39	26.45 ± 0.29	30.85 ± 0.87	42.69
Top-k, $512 \to 20$	96.09	34.95 ± 0.51	76.80 ± 0.61	47.64 ± 0.33	27.34 ± 0.20	31.72 ± 1.05	43.69
S^2-JEPA (ours)							
Probe-guided	96.17	36.45 ± 1.33	$\underline{79.41 \pm 0.31}$	$\mathbf{53.40 \pm 0.67}$	$\mathbf{29.72 \pm 0.33}$	$\mathbf{36.73 \pm 0.02}$	**47.14**
Label-free	96.02	$\mathbf{37.34 \pm 1.23}$	$\mathbf{79.78 \pm 0.65}$	$\underline{52.55 \pm 1.29}$	$\underline{29.40 \pm 0.34}$	$\underline{34.71 \pm 1.22}$	$\underline{46.76}$
Reference-free	95.91	$\underline{36.60 \pm 0.77}$	78.74 ± 0.50	50.61 ± 0.73	28.73 ± 0.15	33.52 ± 1.29	45.64

Table A15: **Transfer of encoder features.** Frozen linear-probe accuracy (%) on five downstream datasets after 100 CIFAR-100 pretraining epochs. Sparsity is measured on the pretrained projector's CIFAR-100 evaluation outputs. Mean is the average over the five datasets. Bold and underline rank methods with at least 90% projector sparsity.

D.6 Representation quality

Table A18 reports retrieval results on STL-10. Reference-free S^2-JEPA matches dense kNN@1 accuracy and is one point lower on kNN@10 while using 13× less storage. At similar sparsity, its kNN accuracy is 8–9 points higher than the fixed target, and its codes have the highest purity. Table A17 measures dependence between projector coordinates using normalized HSIC. S^2-JEPA has lower coordinate dependence than all dense baselines and fixed Rectified LpJEPA targets. The NCL variants reach similar dependence, but only 182 and 293 of their 512 coordinates are non-constant, respectively, whereas S^2-JEPA uses all 512. Compressing the dense codes instead does not reach the same point (Tab. A14). PCA to 20–80 dimensions lowers kNN@10 accuracy by 4.5–4.8 points, and int8 quantization keeps accuracy but needs about 6× more storage than S^2-JEPA.

Method / target	Projector sparsity (%) ↑ CIFAR-100 in-domain	DTD	CIFAR-10	Flowers	Food	Pets	Avg.	Transfer acc. ↑
Dense self-supervised baselines								
SimCLR	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00	46.76
VICReg	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00	48.97
LeJEPA	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00	43.04
Rectified LpJEPA								
$\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$	44.33 ± 0.09	48.71 ± 0.60	39.86 ± 0.21	36.61 ± 0.62	34.48 ± 0.31	35.54 ± 0.07	39.04	45.18
$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	95.49 ± 0.04	94.64 ± 0.06	95.47 ± 0.05	95.56 ± 0.14	95.30 ± 0.11	96.04 ± 0.01	95.40	36.96
$\mathcal{RGN}_2(0, \sigma_{\mathrm{GN}})$	36.21 ± 0.09	41.43 ± 0.88	31.63 ± 0.24	28.30 ± 0.83	25.90 ± 0.04	27.20 ± 0.09	30.89	44.72
Ablation and top-k baselines								
Sparsity pacing only	95.95 ± 0.04	95.81 ± 0.10	95.79 ± 0.01	96.27 ± 0.02	96.08 ± 0.07	96.10 ± 0.05	96.01	42.73
Fixed top-k	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09	42.69
Annealed top-k	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09 ± 0.00	96.09	43.69
S^2-JEPA (ours)								
Probe-guided	96.17 ± 0.03	95.18 ± 0.01	95.71 ± 0.03	96.12 ± 0.18	96.00 ± 0.01	95.98 ± 0.10	95.80	47.14
Label-free, dense reference	96.02 ± 0.04	95.02 ± 0.05	95.49 ± 0.03	96.07 ± 0.14	95.68 ± 0.04	95.79 ± 0.09	95.61	46.76
Reference-free	95.91 ± 0.02	95.05 ± 0.04	95.46 ± 0.01	95.97 ± 0.10	95.60 ± 0.13	95.81 ± 0.05	95.58	45.64

Table A16: **Projector sparsity across downstream datasets.** Percentage of zero entries (↑) on each dataset's test images, using the same frozen CIFAR-100/ResNet-18 checkpoint across all columns. Means and sample standard deviations use three runs, and all models train for 100 epochs. All projector outputs have 512 coordinates. Avg. is the mean sparsity over the five downstream datasets, and Transfer acc. is the mean frozen encoder accuracy over the same datasets from Tab. A15.

Table A17: **Pairwise dependence between projector coordinates.** Normalized HSIC on a class-balanced subset of 1000 CIFAR-100 test images after 100 ResNet-18 epochs. The mean covers off-diagonal pairs with defined bandwidths and positive self-HSIC; valid coordinates are non-constant on the subset. Values are means and sample standard deviations over three runs, with nHSIC scaled by 10^3. Sparsity is measured on the same subset. Encoder and projector columns report frozen linear-probe accuracy (%) on the full CIFAR-100 test set, as in Tab. 1.

Method / target	Subset sparsity (%) ↑	Valid coordinates of 512	10^3 nHSIC ↓	Encoder ↑	Projector ↑
Dense self-supervised baselines					
SimCLR	0.00 ± 0.00	512.00 ± 0.00	12.24 ± 0.22	60.70	54.96
VICReg	0.00 ± 0.00	512.00 ± 0.00	30.95 ± 0.43	63.04	55.64
LeJEPA	0.00 ± 0.00	512.00 ± 0.00	16.91 ± 0.14	54.87	44.88
Nonnegative self-supervised baselines					
NCL-ReLU	95.39 ± 0.07	181.67 ± 8.39	8.74 ± 0.11	52.90	46.55
NCL-RepReLU	96.10 ± 0.19	293.00 ± 21.07	7.92 ± 0.20	53.88	42.33
Rectified LpJEPA, $p = 1$					
$\mathcal{RGN}_1(1.0, \sigma_{\mathrm{GN}})$	9.86 ± 0.07	512.00 ± 0.00	16.78 ± 0.07	57.83	46.72
$\mathcal{RGN}_1(0.5, \sigma_{\mathrm{GN}})$	21.91 ± 0.34	512.00 ± 0.00	17.08 ± 0.09	58.19	48.35
$\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$	43.83 ± 0.06	512.00 ± 0.00	18.21 ± 0.06	57.24	48.13
$\mathcal{RGN}_1(-0.5, \sigma_{\mathrm{GN}})$	65.54 ± 0.23	512.00 ± 0.00	19.48 ± 0.08	56.01	47.24
$\mathcal{RGN}_1(-1.0, \sigma_{\mathrm{GN}})$	84.05 ± 0.04	512.00 ± 0.00	17.60 ± 0.08	54.17	43.95
$\mathcal{RGN}_1(-1.5, \sigma_{\mathrm{GN}})$	91.46 ± 0.02	512.00 ± 0.00	12.91 ± 0.04	52.68	41.21
$\mathcal{RGN}_1(-2.0, \sigma_{\mathrm{GN}})$	94.31 ± 0.02	512.00 ± 0.00	10.35 ± 0.01	49.14	37.31
$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	95.51 ± 0.03	511.67 ± 0.58	9.56 ± 0.10	44.56	32.28
$\mathcal{RGN}_1(-3.0, \sigma_{\mathrm{GN}})$	96.30 ± 0.02	511.00 ± 1.00	9.38 ± 0.18	35.31	23.79
Rectified LpJEPA, $p = 2$					
$\mathcal{RGN}_2(0, \sigma_{\mathrm{GN}})$	35.62 ± 0.14	512.00 ± 0.00	18.07 ± 0.12	57.58	48.16
$\mathcal{RGN}_2(-2.0, \sigma_{\mathrm{GN}})$	94.24 ± 0.07	505.33 ± 7.23	10.48 ± 0.05	38.27	27.59
$\mathcal{RGN}_2(-2.25, \sigma_{\mathrm{GN}})$	95.46 ± 0.02	490.00 ± 1.00	10.00 ± 0.13	31.80	22.56
$\mathcal{RGN}_2(-2.5, \sigma_{\mathrm{GN}})$	96.14 ± 0.16	497.33 ± 9.45	10.45 ± 0.22	27.19	18.76
S^2-JEPA (ours)					
Probe-guided	96.21 ± 0.02	512.00 ± 0.00	7.61 ± 0.13	60.75	48.83
Label-free, dense reference	96.04 ± 0.02	512.00 ± 0.00	7.95 ± 0.12	60.62	48.38
Reference-free	95.92 ± 0.01	512.00 ± 0.00	8.40 ± 0.08	58.97	46.73

Table A18: **Projector retrieval on STL-10.** Evaluation sparsity, active coordinates per image, kNN accuracy with a $7200/800$ gallery/query split, purity@64 (defined in Tab. A5), coordinates never active, and bytes per image (fp16 values, uint16 indices, and an int32 row pointer for sparse outputs; fp16 values for dense outputs). All outputs have 512 coordinates. Accuracy, sparsity, and purity are percentages.

Representation	Sparsity (%) ↑	Active	kNN@1 ↑	kNN@10 ↑	Purity ↑	Unused ↓	Bytes ↓
$\mathcal{RGN}_1(0, \sigma_{\mathrm{GN}})$	42.37	295.1	69.54 ± 1.92	74.58 ± 0.59	12.66 ± 0.04	0.0	1024.0
$\mathcal{RGN}_1(-2.5, \sigma_{\mathrm{GN}})$	96.12	19.9	61.46 ± 0.44	64.96 ± 0.52	56.56 ± 0.72	0.0	83.5
S^2-JEPA, reference-free	96.34	18.8	69.63 ± 0.98	73.54 ± 0.69	59.96 ± 2.54	0.0	79.0