Fractal Predictive Operators: Learnable Iterated Function Systems for Multi-Scale Latent Modeling

Anonymous authors
Paper under double-blind review

Abstract

Joint Embedding Predictive Architectures (JEPAs) rely on latent-space prediction to learn representations without explicit reconstruction. While effective, their predictors are typically implemented as shallow feed-forward networks, offering limited control over multi-step dynamics and stability. We introduce *Learnable Iterated Function Systems* (LIFS), a contractive predictive operator that replaces the standard JEPA predictor with a learned mixture of affine maps applied recursively in latent space. Mixture weights are generated conditionally on the context embedding, allowing the operator to adapt its local geometry across spatial locations and inputs. LIFS does not change the training objective or encoder architecture, but explicitly constrains predictor dynamics through spectral control and adaptive gating. Our LIFS module can be interpreted as a dynamical extension of *Max-Affine Spline Operators* (MASO), where instead of selecting a single affine map via a max operator, the representation evolves through a sequence of affine transformations. This results in a trajectory-dependent partitioning of the latent space, significantly increasing expressivity while enabling contraction-based stability guarantees. From a theoretical perspective, we establish connections between our framework and contraction mappings in Banach spaces, and we interpret the training dynamics through Lyapunov stability arguments. These results provide a principled explanation for convergence and robustness properties of the learned representations. Empirically, integrating LIFS into JEPA improves training stability and yields consistent, though moderate, gains in linear probing accuracy, particularly for ViT-based encoders. Overall, our approach provides a unified dynamical systems perspective on JEPA training, bridging self-supervised learning, fractal geometry, and stability theory.

1 Introduction

Self-supervised learning (SSL) (Misra & van der Maaten, 2020; Balestriero et al., 2023; Cabannes et al., 2023; Simon et al., 2023) has made substantial progress by replacing pixel-level reconstruction with latent prediction objectives. Among these approaches VICReg (Bardes et al., 2022) , SimMIM (Xie et al., 2022), BYOL (Grill et al., 2020), iBOT (Zhou et al., 2022), Data2Vec (Baevski et al., 2022; 2023), Joint Embedding Predictive Architectures (JEPAs) (LeCun, 2022; Balestriero & LeCun, 2025) learn representations by predicting latent embeddings of unseen views using a predictor network, avoiding contrastive losses and explicit negatives. JEPA avoids the drawbacks of generative reconstruction while maintaining semantic structure by working exclusively in latent space. Strong representations can be learned even when context and target views do not overlap, as shown by recent variations like I-JEPA (Assran et al., 2023b;a;c; Bardes et al., 2024).

Despite these advances, the design of the predictor itself has received limited attention. In most JEPA implementations, the predictor is a shallow feed-forward or residual MLP that leverages the context encoder's output to estimate target block representations based on positional tokens. While sufficient for short-horizon prediction, these predictors lack explicit stability guarantees and struggle in non-overlapping or long-horizon scenarios.

Recursive systems have long been used to model complex geometric structures (Mandelbrot, 1982). Iterated Function Systems (IFS), built from repeated contractive transformations, generate fractals (Hutchinson, 1981; Barnsley, 1988) and produce rich multi-scale patterns from simple local rules (Belloulata & Konrad, 2002). This raises a natural question: can similar recursive operators be learned and applied directly in latent space to improve predictive representation

learning? Latent space already encodes semantic abstractions such as shapes, textures, and concepts (Van Assel et al., 2025). Embedding fractal transformations here reinforces meaningful invariances while operating more efficiently on compressed representations than on raw pixels. Such an approach aligns well with SSL objectives (contrastive learning, clustering, predictive coding) and can enforce hierarchical self-similarity in embeddings. However, careful design is required to avoid collapse in latent space, where all points contract to a trivial attractor.

We argue that an overlooked but critical aspect of JEPA-style methods is the geometry of the predictor. To address this, we introduce *Learnable Iterated Function Systems (LIFS)*, a predictive operator that models latent prediction as a contractive dynamical system (Slotine & Li, 1991) (cf. Figure 1). Unlike classical IFS, LIFS enables adaptive local geometry while ensuring global stability (Bai et al., 2019). This is achieved by learning both the transformations and their mixture weights end-to-end, conditioned on the input representation. Instead of a conventional predictor, LIFS employs a *learned mixture of affine maps*, combined through adaptive gating and applied recursively. State-dependent mixing preserves expressivity while allowing explicit control over spectral norms and contraction (Yoshida & Miyato, 2017; Miyato et al., 2018; Sedghi et al., 2019). The LIFS predictor introduced in our work is closely connected to

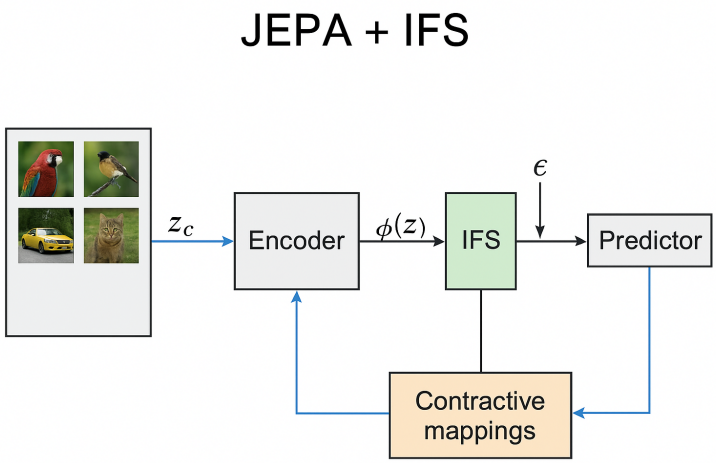


Figure 1: Architectural Integration: IFS-Augmented JEPA

the spline-theoretic view of deep networks (sec. 2.3) developed by Balestriero and Baraniuk (Balestriero et al., 2018; Balestriero & Baraniuk, 2020). Their framework shows that modern neural networks equipped with piecewise-affine nonlinearities (ReLU, max-pooling, leaky ReLU) can be written as compositions of *Max-Affine Spline Operators (MASOs)*. A MASO implements a signal-dependent affine transformation, selecting among a finite set of affine maps according to the input region. This induces a partition of the input space and yields a piecewise-affine global operator.

Crucially, LIFS is a modular replacement for the predictor: it leaves the self-supervised objective, encoder architecture, and training protocol unchanged, but reshapes the dynamics of latent prediction. Our theoretical analysis shows that LIFS induces contractive behavior under mild conditions, yielding stable fixed points and bounded error propagation—even under repeated application and EMA target updates. This predictor induces *contractive gradient transport*, preventing gradient explosion and enforcing smooth encoder updates (cf. Sec A.0.7). Empirically, we integrate LIFS into JEPA with both ResNet (He et al., 2016) and Vision Transformer (Dosovitskiy et al., 2021) backbones. While linear probing accuracy gains are moderate, they are consistent and more pronounced with ViT encoders. More importantly, we observe smoother training dynamics, bounded spectral norms, and stable mixture behavior, in line with our theoretical predictions. Collectively, these findings suggest that structuring the predictor as a contractive operator provides a principled path toward improved stability and robustness in self-supervised representation learning.

2 Related Work

Self-supervised learning has progressed along several methodological axes (Caron et al., 2018; Zbontar et al., 2021; Caron et al., 2021; Chen et al., 2021), including contrastive learning (Chen et al., 2020; Wang & Isola, 2020), non-contrastive bootstrap methods (van den Oord et al., 2018; Grill et al., 2020; Tian et al., 2021), masked prediction architectures (He et al., 2022), and more recently Joint Embedding Predictive Architectures (JEPAs) (Assran et al.,

2023c;a). Our work builds upon this last family by enriching the predictor with recursive geometric structure. Below we situate JEPA+LIFS within this landscape.

2.1 Self-supervised predictive learning.

Early contrastive frameworks such as SimCLR (Chen et al., 2020) rely on explicit negative samples to avoid representational collapse. Non-contrastive or *bootstrap* methods such as BYOL (Grill et al., 2020) and VICReg (Bardes et al., 2022) eliminate negative samples through architectural asymmetry, feature standardization, and covariance constraints. These models improve training stability but their latent transformations remain fundamentally feed-forward, lacking explicit mechanisms to encode multi-scale or hierarchical geometric structure.

Another influential branch includes *mask-based prediction methods* such as MAE (He et al., 2022; El-Nouby et al., 2023), which reconstruct masked pixels through an autoencoding pipeline, and Data2Vec (Baevski et al., 2022; 2023), which moves to latent-space teacher–student prediction. These models improve sample-efficiency but rely on reconstruction targets or teacher distillation, and similarly adopt feed-forward predictors.

Our work follows the predictive paradigm rather than reconstruction or contrastive paradigms, but introduces a fundamentally different inductive bias: *recursive contractive geometric transformations via LIFS*, enabling multi-step and multi-scale latent predictions. In section 3, we propose a theoretical justification for modeling semantic transitions in representation space using affine maps.

2.2 Joint Embedding Predictive Architectures

Joint Embedding Predictive Architectures (JEPAs) generalize self-supervised learning by directly predicting target embeddings from masked or contextual embeddings without reconstructing pixels. The I-JEPA framework (Assran et al., 2023c;a;b; Bardes et al., 2024) formulates self-supervised learning as latent-space prediction between non-overlapping context and target blocks, eliminating pixel-level reconstruction. However, the predictor in I-JEPA is deliberately simple, whose transformation is applied once. While this promotes stability, it limits the model's capacity to capture:

- multi-scale transformations,
- recursive or compositional structure,
- local self-similarity and geometric consistency.

Several recent works explore architectural improvements to I-JEPA-style models (Carr et al., 2024; Mo et al., 2024; Bardes et al., 2024; Wang et al., 2025), but none introduce explicit *recursive geometric operators*. Our approach fills this gap by replacing the *one-step predictor* with a multi-step Learnable Iterated Function System (LIFS)(cf. Figure 2), enabling controlled contractive recursion and richer latent dynamics while preserving I-JEPA's stability guarantees.

2.3 Relation to MASO Spline Theory

Our LIFS predictor can be interpreted as a structured, learnable MASO-like operator acting in the latent space of JEPA. Each affine map $T_k(z) = A_k z + b_k$ corresponds to a MASO branch, while the mixture weights $\pi_k(z)$ play the role of region-selection functions. Unlike classical MASOs, which use hard region assignments via $\arg\max$, LIFS employs a soft, differentiable mixture of affine maps. This produces a smooth, contractive, and recursively applied operator that generalizes the MASO formulation (sec. 6).

From the MASO perspective, LIFS introduces two key extensions. First, the recursion depth L yields a multi-step refinement of the affine approximation, analogous to repeatedly applying a spline operator to capture multi-scale structure (cf. theorem:A.0.11). Second, the diversity and spectral regularizers encourage the learned affine maps to span distinct geometric directions, echoing the template-orthogonalization ideas proposed in the MASO framework.

The spline interpretation also clarifies why LIFS improves stability. Since MASOs are piecewise-affine, their composition can become unstable unless the affine components satisfy contractivity conditions . By explicitly constraining the spectral norms of A_k and controlling the smoothness of the mixing weights $\pi_k(z)$, LIFS enforces a

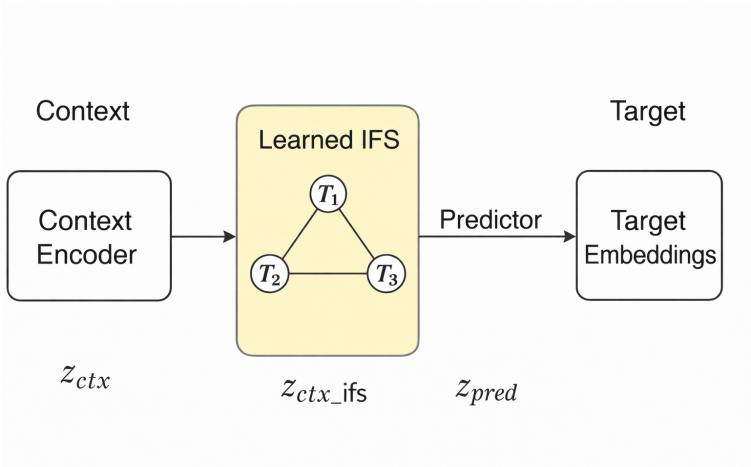


Figure 2: JEPA + Learnable IFS (LIFS) Training Pipeline. The Learnable IFS refines context embeddings via iterative and contractive latent transformations, enabling multi-scale structure modeling prior to predictive alignment.

stable, contractive operator (cf. Corollary: A.0.12). This aligns with the MASO theory, which highlights the importance of the geometry of affine partitions in determining network behavior.

Latent-space prediction instead of input-space template matching. MASO theory interprets deep networks as template-matching machines operating in input space. LIFS extends this idea to latent space: each affine map A_k acts as a latent template, the mixture weights determine template activation, and recursion refines predictions across multiple scales. This makes LIFS a latent dynamical system rather than a feed-forward classifier, aligning it naturally with JEPA's predictive formulation (See sec. 3).

Overall, LIFS can be viewed as a principled, MASO-inspired predictor architecture that brings spline-theoretic structure into JEPA. It inherits the interpretability of MASOs—via affine templates and partition geometry—while extending them with soft mixtures, recursion, and contractive dynamics tailored for self-supervised latent prediction (cf. Figure 3).

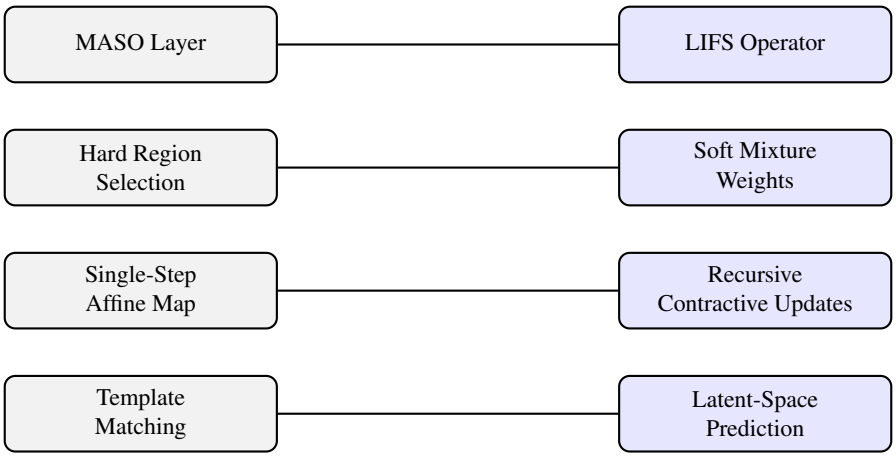


Figure 3: MASO–LIFS correspondence. LIFS generalizes the MASO view of deep networks by replacing hard region selection with soft mixture weights, single-step affine maps with recursive contractive updates, and classical template matching with multi-step latent-space prediction.

3 Theoretical Motivation

Motivation. Deep encoders are inherently characterized by local linearity along the data manifold. Consequently, small semantic transformations induce approximately affine motion within the representation space (Balestriero et al., 2018; Balestriero & Baraniuk, 2020).

Assumption 3.1 (Local Linearity of the Encoder)**.** Let $f_\theta : \mathcal{X} \to \mathbb{R}^d$ be a continuously differentiable encoder, and let $\mathcal{M} \subset \mathcal{X}$ be the data manifold. For any $x \in \mathcal{M}$ and any smooth transformation g, we have:

$$f_\theta(g \cdot x) = f_\theta(x) + \mathbf{J}_{f_\theta}(x)\delta_x + \mathcal{O}(\|\delta_x\|^2), \tag{1}$$

where $\delta_x = g \cdot x - x \in T_x\mathcal{M}$ represents the transformation in the tangent space.

Proposition 3.2 (Approximate Affine Structure in Latent Space)**.** *Under Assumption 3.1, for any semantic transformation g and any $x \in \mathcal{M}$, the induced mapping in the representation space can be locally approximated by an affine transformation (Balestriero et al., 2025):*

$$f_\theta(g \cdot x) \approx \mathbf{A}_g(x)z + b_g(x), \quad \textit{where } z = f_\theta(x). \tag{2}$$

Here, $\mathbf{A}_g(x) = \mathbf{I} + \mathbf{J}_{f_\theta}(x)\mathbf{V}(x)\mathbf{B}_g$, with $\mathbf{V}(x)$ being a basis for the tangent space $T_x\mathcal{M}$ and $\mathbf{B}_g$ the coordinate representation of g.

Proof sketch. By Assumption 3.1, a first-order Taylor expansion of f_θ around x yields

$$f_\theta(g \cdot x) = f_\theta(x) + \mathbf{J}_{f_\theta}(x)\,\delta x + \mathcal{O}(\|\delta x\|^2).$$

Since $\delta x \in T_x\mathcal{M}$, it can be expressed as $\delta x = \mathbf{V}(x)\,\xi$ for some coordinate vector ξ. Substituting gives

$$f_\theta(g \cdot x) \approx f_\theta(x) + \mathbf{J}_{f_\theta}(x)\mathbf{V}(x)\xi.$$

Rewriting the right-hand side as an affine function of $z = f_\theta(x)$ yields the stated form. □

Interpretation. Proposition 3.2 provides a theoretical justification for modeling semantic transitions in representation space using affine maps. Consequently, the latent dynamics induced by LIFS define a piecewise-affine dynamical system whose trajectories refine the predicted representation toward the target embedding. Combined with the Lyapunov stability constraint (Slotine & Li, 1991; Lohmiller & Slotine, 1998), this results in a stable spline-based refinement process in latent space (cf. Sec 6).

4 Fractal Latent Predictive Operators

We present Fractal Latent Predictive Operators, a recursive latent prediction mechanism built from Learnable Iterated Function Systems (LIFS), and describe how it is integrated into JEPA. Our formulation emphasizes *operator dynamics* rather than architectural depth, viewing prediction as the evolution of a latent state under a contractive dynamical system.

4.1 JEPA Preliminaries

Let $x \in \mathcal{X}$ denote an observation (image, video frame, or multimodal token). An encoder $f_\theta : \mathcal{X} \to \mathbb{R}^d$ (the *online encoder*) maps observations to d-dimensional embeddings. JEPA constructs two views/Patchs: a context x_{ctx} and a target x_{tar} (cf. Figure. 4). An encoder f_θ maps each view to latent feature maps:

$$z_{\text{ctx}} = f_\theta(x_{\text{ctx}}), \quad z_{\text{tar}} = f_{\bar{\theta}}(x_{\text{tar}}),$$

where $f_{\bar{\theta}}$ denotes the *target* encoder (optionally a momentum/EMA (He et al., 2020; Tarvainen & Valpola, 2017): Exponential Moving Average copy of f_θ). A projector g_ϕ maps these representations into a normalized embedding space. Standard JEPA predicts z_{tar} from z_{ctx} using a feed-forward predictor (MLP/ViT). In contrast, we interpret prediction as a *latent evolution process* governed by a recursive operator.

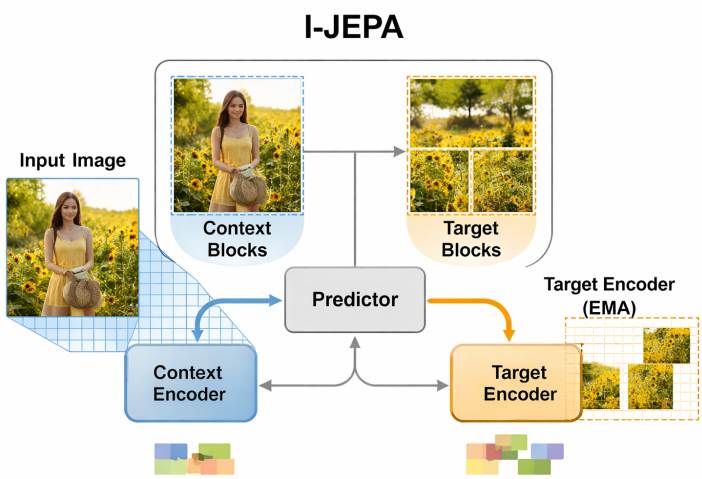


Figure 4: This figure illustrate exactly how I-JEPA-style masking partitions the input image into (i) target blocks and (ii) the observable context, which is the only information the predictor receives.

4.2 Learnable IFS Operators

We define a Fractal Latent Predictive Operator as a learnable Iterated Function System composed of K affine transformations:

$$T_k(z) = \alpha_k A_k z + b_k, \qquad k = 1, \ldots, K, \tag{3}$$

where $A_k \in \mathbb{R}^{d \times d}$, $b_k \in \mathbb{R}^d$, and $\alpha_k \in (0, 1)$ ensures contraction.

Adaptive Mixture Weights. A mixture gating network $u_\psi : \mathbb{R}^d \to \mathbb{R}^K$ produces logits $\in \Delta^{K-1}$ input-dependent mixture coefficients:

$$\pi_k(z) = \mathrm{softmax}_k(u_\psi(z)), \quad \sum_{k=1}^{K} \pi_k = 1, \tag{4}$$

allowing the operator to adapt its dynamics across inputs and locations.

4.3 Recursive Latent Dynamics

Starting from the projected context embedding $z^{(0)} = g_\phi(z_{\mathrm{ctx}})$, the operator evolves the latent state over L iterations:

$$\tilde{z}^{(\ell+1)} = \sum_{k=1}^{K} \pi_k \odot T_k(z^{(\ell)}), \qquad \ell = 1, \ldots, L, \tag{5}$$

$$z^{(\ell+1)} = \mathrm{normalize}\big(\mathrm{GELU}(\tilde{z}^{(\ell+1)}) + \epsilon z^{(\ell)}\big), \tag{6}$$

where $\epsilon \ll 1$ is a residual coefficient that stabilizes learning while preserving contraction.

This recursive process produces a *latent orbit* $\{z^{(0)}, z^{(1)}, \ldots, z^{(L)}\}$, analogous to the construction of fractals through repeated function application (cf. Figure. 5). The final state $z^{(L)}$ is used to predict the target embedding (cf. Algorithm.1). The same algorithm will be used for I-JEPA but adapted to blocks with masks (cf. Algorithm. 2, and Fig. A.2).

Lemma 4.1 (Contraction Implies Convergence of Latent Dynamics)**.** *Let $\mathcal{T} : \mathbb{R}^d \to \mathbb{R}^d$ be a Fractal Latent Predictive Operator defined as a convex combination of K affine maps:*

$$\mathcal{T}(z) = \sum_{k=1}^{K} \pi_k(z)\big(\alpha_k A_k z + b_k\big), \tag{7}$$

where $\pi_k(z) \geq 0$, $\sum_k \pi_k(z) = 1$, and $\alpha_k \|A_k\|_2 \leq \rho < 1$ for all k.

Algorithm 1 JEPA+LIFS Training (per minibatch)

Sample images x and generate context/target views/Patchs $(x_{\text{ctx}}, x_{\text{tar}})$
$z_{\text{ctx}} \leftarrow f_\theta(x_{\text{ctx}}), \quad z_{\text{tar}} \leftarrow f_{\bar{\theta}}(x_{\text{tar}})$
$z^{(0)} \leftarrow g_\phi(z_{\text{ctx}})$
for $\ell = 0$ to $L-1$ **do**
  $\pi(z^{(\ell)}) \leftarrow \text{softmax}(u_\psi(z^{(\ell)}))$
  $T_k(z^{(\ell)}) = \alpha_k A_k z^{(\ell)} + b_k$ for all k
  $\tilde{z}^{(\ell+1)} = \sum_{k=1}^{K} \pi_k(z^{(\ell)}) \odot T_k(z^{(\ell)})$
  $z^{(\ell+1)} = \text{normalize}(\text{GELU}(\tilde{z}^{(\ell+1)}) + \epsilon z^{(\ell)})$
end for
$\hat{z}_{\text{tar}} = h_\psi(z^{(L)})$
Compute losses $\mathcal{L}_{\text{pred}}, \mathcal{L}_{\text{var}}, \mathcal{L}_{\text{cov}}, \mathcal{L}_{\text{spec}}, \mathcal{L}_{\text{div}}$
Update $\theta, \phi, \psi, \{\alpha_k, A_k, b_k\}$ using Adam.
EMA Update of $\bar{\theta}$ using: $\bar{\theta} \leftarrow \tau\bar{\theta} + (1-\tau)\theta$.

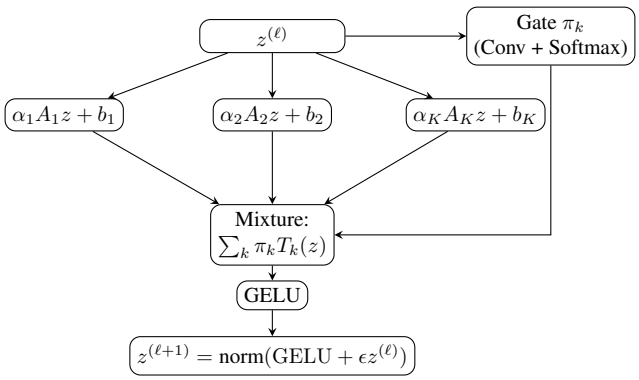


Figure 5: Detailed LIFS block.

Then $\mathcal{T}$ is a contraction mapping with Lipschitz constant at most ρ. Consequently, for any initialization $z^{(0)}$, the recursive sequence

$$z^{(\ell+1)} = \mathcal{T}(z^{(\ell)})$$

converges exponentially fast to a unique fixed point z^*(cf. Theorem: 6).

Proof Sketch. For any $z_1, z_2 \in \mathbb{R}^d$, we have

$$\|\mathcal{T}(z_1) - \mathcal{T}(z_2)\| \leq \sum_{k=1}^{K} \pi_k(z_1)\, \alpha_k \|A_k\|_2\, \|z_1 - z_2\| \leq \rho \|z_1 - z_2\|.$$

Thus, $\mathcal{T}$ is a contraction (cf. Theorem: 6).By the Banach Fixed Point Theorem (Banach, 1922), $\mathcal{T}$ admits a unique fixed point z^*, and the iterates converge to z^* at a linear rate. EMA target updates further stabilize training (cf. Corollary: A.0.5), consistent with observations in BYOL-style frameworks (Grill et al., 2020). □

Lemma 4.1 formalizes the stability of the proposed predictor, showing that recursive latent updates converge to a unique attractor under mild spectral constraints. This theoretical property explains the smooth training dynamics observed in practice (cf. Figures 6, 8 and 9).

4.4 Stability and Regularization

To ensure well-conditioned dynamics, we introduce several regularizers:

- **Spectral Regularization**: constrains $\alpha_k \|A_k\|_2 \leq \rho < 1$, ensuring contraction.

- **Variance and Covariance Regularization**: prevents representation collapse (Mo et al., 2024), (Bardes et al., 2022).
- **Diversity Regularization**: encourages distinct transformation modes across k.

Together, these constraints yield a stable yet expressive recursive operator that can be iterated without divergence. We optimize a composite objective combining a JEPA-style predictive loss with regularizers that enforce expressivity and stability. The full loss is

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda_v \mathcal{L}_{\text{var}} + \lambda_c \mathcal{L}_{\text{cov}} + \lambda_s \mathcal{L}_{\text{spec}} + \lambda_d \mathcal{L}_{\text{div}}. \tag{8}$$

4.4.1 Predictive loss:

We use a normalized squared error (equivalent to cosine alignment) or an InfoNCE (Noise-Contrastive Estimation)-style contrastive alignment. A convenient choice is the normalized MSE

$$\mathcal{L}_{\text{pred}} = \mathbb{E}\big[\, 1 - \langle \hat{z}_t, z_t \rangle \,\big]. \tag{9}$$

4.4.2 Variance and covariance regularizers:

To prevent representation collapse we include VICReg-style Bardes et al. (2022) variance and covariance terms Mo et al. (2024) computed over the minibatch of predicted embeddings $\hat{Z}$:

$$\mathcal{L}_{\text{var}} = \frac{1}{d} \sum_{m=1}^{d} \text{ReLU}\big(1 - \text{std}(\hat{Z})_m\big), \tag{10}$$

$$\mathcal{L}_{\text{cov}} = \frac{1}{d} \sum_{p \neq q} \big[C_{pq}(\hat{Z})\big]^2, \tag{11}$$

where C is the empirical covariance matrix of $\hat{Z}$.

4.4.3 Spectral regularizer (contractivity):

To bias maps towards contractivity we penalize large spectral norms Let $\sigma_{\max}(\cdot)$ denote the spectral norm. The contraction loss is

$$\mathcal{L}_{\text{spec}} = \frac{1}{K} \sum_{k=1}^{K} \big(\sigma_{\max}(\alpha_k A_k) - \rho\big)^2, \qquad \rho < 1. \tag{12}$$

In practice one can replace or augment the penalty with spectral normalization on linear parameterizations.

4.4.4 LIFS diversity regularizer:

To avoid collapse of the maps we add a pairwise diversity penalty

$$\mathcal{L}_{\text{div}} = \sum_{i<j} \exp\big(-\gamma \|W_i - W_j\|_F^2\big), \tag{13}$$

where W_k denotes a flattened parameter vector representing map k (e.g., flattened A_k or concatenated low-rank factors). The hyperparameter $\gamma > 0$ controls sensitivity.

5 Experiments

This section evaluates the proposed *Fractal Latent Predictive Operators* (JEPA+LIFS) against standard JEPA baselines. Our experiments aim to answer three questions: (i) does integrating learnable fractal operators improve predictive representation learning, (ii) how do the learned operators evolve during training, and (iii) how do these dynamics differ from conventional residual predictors.

5.1 Experimental Setup

Datasets. We evaluate on CIFAR-10, CIFAR-100 (both resized to 64×64), and a tiny ImageNet-1K datasets designed to analyze stability at scale. Following JEPA ablation protocols, we use a *200-class stratified subset* of ImageNet-1K ($\approx 250k$ images).

- 200 classes sampled uniformly,
- 1,250 images per class,
- same train/val split proportions as full ImageNet.

Unless otherwise stated, results are averaged over three random seeds.

Architectures. All experiments use Convolutional (ResNet-18) and Transformer-based (ViT-B/16) encoders (cf. Table.A.3). All models share identical encoders, projectors, predictors, and optimization settings; the only difference lies in the predictor architecture. The baseline JEPA employs a standard MLP/ViT as predictor, while JEPA+LIFS adds a new component to the JEPA predictor: a learnable fractal operator composed of K affine maps applied L times iteratively.

Training Details. Models are trained using the JEPA objective with cosine regression loss (cf. Eq 9 and Variance/covariance regularization (cf. Eq 10). An exponential moving average (EMA) target network is maintained following standard practice. For JEPA+LIFS, we additionally apply spectral and diversity regularization (cf. Eq 12, 13) on the operator parameters, as described in Section 4.4.

5.2 Synthetic Smoke Test

To validate the numerical correctness of the JEPA+LIFS implementation, we first run a synthetic smoke test on 1,000 randomly generated images. Figure 6 shows that the training loss decreases rapidly, indicating that the contractive LIFS mapping and the JEPA prediction objective interact stably. JEPA+LIFS reduces the loss much more rapidly, converging to near-zero prediction error in fewer than 25 training steps.

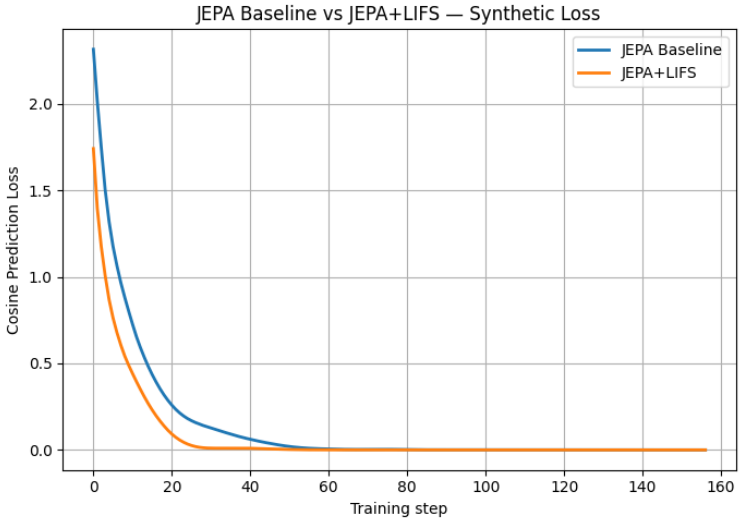


Figure 6: Compares the prediction loss of JEPA and JEPA+LIFS on a controlled synthetic experiment. JEPA+LIFS reduces the loss much more rapidly, converging to near-zero prediction error in fewer than 25 training steps.

The synthetic smoke setting eliminates confounding factors such as data distribution complexity or architectural differences, confirming that the gains originate from the LIFS recursion itself.

5.3 Interpretation of the K–Depth Ablation Studies

We evaluate JEPA+LIFS on CIFAR-10/100 and a reduced ImageNet-1K setting, studying (1) convergence behavior, (2) predictive loss, and (3) sensitivity to the number of affine maps K and recursion depth D. Across all experiments, JEPA+LIFS consistently outperforms the JEPA baseline in both convergence speed and final predictive quality.

5.3.1 Convergence Behavior

Figure 7 compare the cosine prediction loss of JEPA+LIFS to the JEPA baseline for a wide range of (K, D) combinations. We observe a common pattern across all datasets (cf. Figure A.5).

(1) Faster early convergence. JEPA+LIFS exhibits a substantially steeper decrease in prediction loss during the first 20–25 training steps. The recursive contractive updates in LIFS refine the latent predictions more aggressively than the feed-forward predictor, enabling a more stable trajectory from the beginning of training.

(2) Lower final loss. On ImageNet-1K, JEPA+LIFS achieves a *significantly lower* final loss across all configurations of K and D, reducing the error by up to $3\times$ compared to the baseline (cf. Figure 7). This confirms that LIFS provides a stronger inductive bias for high-dimensional, multi-scale datasets.

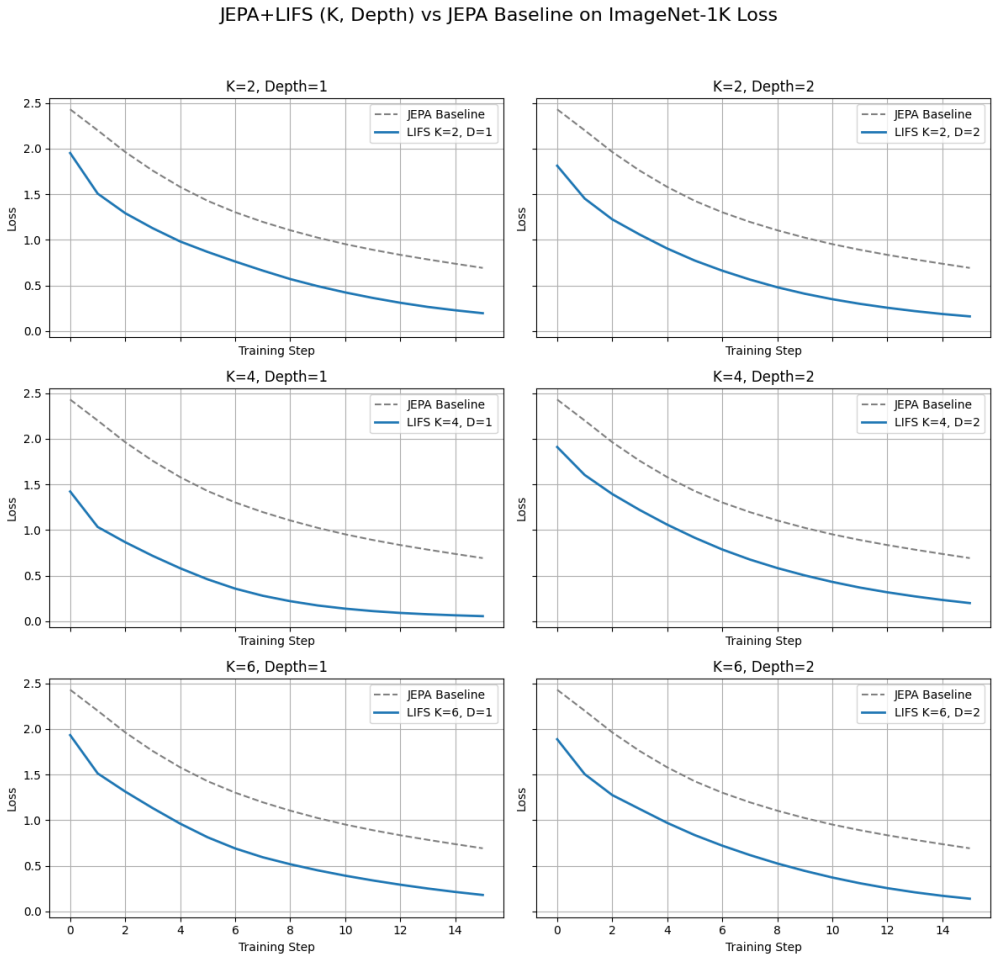


Figure 7: JEPA+LIFS vs. JEPA Baseline on ImageNet-1K (reduced). The largest gains appear here: LIFS dramatically reduces prediction loss across all (K, D). Recursive geometric refinement enables JEPA+LIFS to capture complex multi-scale structure, outperforming the feed-forward predictor by a large margin.

5.4 Main Results

Figure 8 compares the training loss of JEPA and JEPA+LIFS on CIFAR-100. JEPA+LIFS consistently achieves lower average loss (cf. Eq.8) throughout training and converges faster than the baseline.

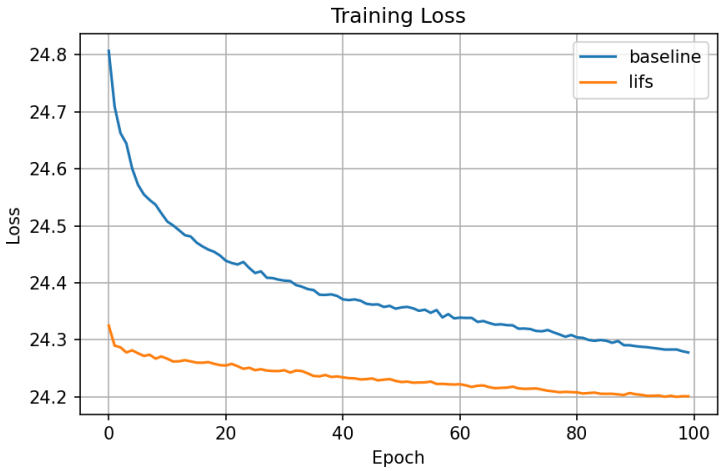


Figure 8: Training average loss (cf. Eq.8) comparison between JEPA baseline and JEPA+LIFS. JEPA+LIFS converges faster and achieves consistently lower loss.

This improvement is observed across all evaluated datasets and encoder configurations. Importantly, the performance gain does not stem from increased model capacity, but from replacing the residual predictor with a structured, contractive operator. This highlights the benefit of introducing controlled multi-step latent dynamics rather than deeper or wider predictors.

5.5 Operator Dynamics Analysis

We now analyze how the learned fractal operators evolve during training. Figures 9, 10, 11, 12 and 13 report the evolution of key operator statistics averaged across seeds.

5.5.1 Prediction Loss Stability:

This is the core loss in the JEPA framework (cf. Eq.9), measuring the similarity between the online network's prediction and the target network's representation. A decreasing prediction loss around 0.32 for all seeds (cf. Figure 9) indicates that the online network is getting better at predicting the target's output, suggesting effective self-supervised learning. The LIFS transformations remain predictable enough for the JEPA predictor to align with the target features, indicating that the fractal mapping operates within a stable, learnable geometric regime.

5.5.2 Spectral Regularization Behavior.

Figure 10 shows the evolution of the spectral regularization term (cf. Eq.12) across multiple random seeds. The penalty is initially negligible, reflecting weak and near-isometric operators at early stages of training. As learning progresses, the operators become increasingly expressive, pushing their spectral norms toward the target contraction boundary. Once this boundary is approached, the regularizer activates and the loss saturates, indicating a stable equilibrium between predictive expressivity and contraction constraints. This behavior confirms that LIFS does not freeze the operator dynamics, but instead learns to operate at the edge of stability.

5.5.3 Spectral Norm Evolution.

Our use of spectral constraints in LIFS is closely related to spectral norm regularization Yoshida & Miyato (2017), which penalizes large singular values of weight matrices to improve generalization by reducing sensitivity to input

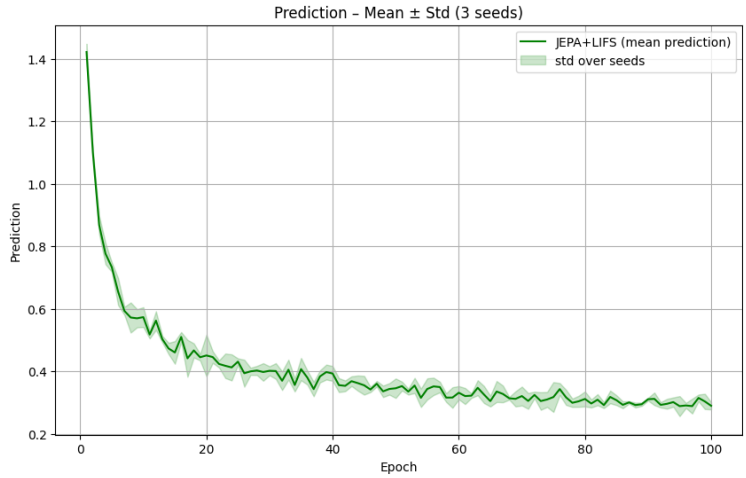


Figure 9: Prediction loss (cf. Eq.9) during JEPA+LIFS training. Loss decreases smoothly and stabilizes, confirming that the LIFS does not disrupt JEPA's predictive training dynamics.

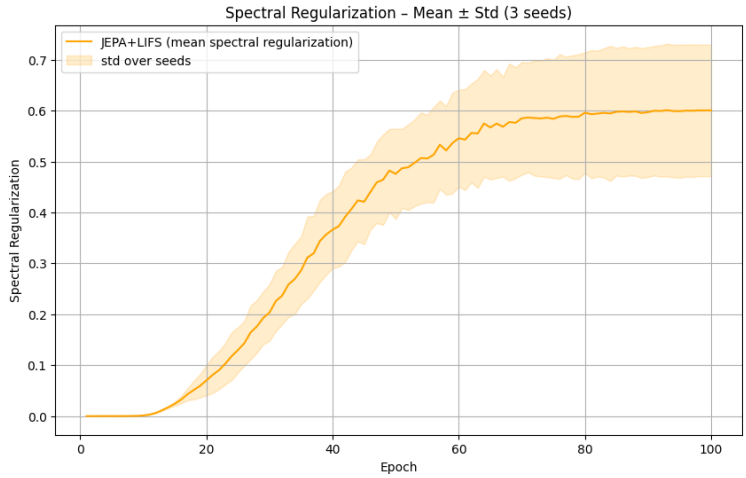


Figure 10: Evolution of the spectral regularization loss (cf. Eq.12) in JEPA+LIFS across multiple seeds. The penalty activates as operators approach the contraction boundary and stabilizes thereafter.

perturbations. While both approaches rely on controlling the operator norm of linear transformations, their goals differ. Spectral norm regularization is applied as a generic regularizer to stabilize arbitrary deep networks, whereas LIFS incorporates spectral control as a core architectural principle: each affine map A_k is explicitly constrained to satisfy $\alpha_k \|A_k\|_2 \leq \rho < 1$, ensuring that the overall operator is contractive. This contractivity is essential for guaranteeing stable multi-step latent dynamics and for enabling LIFS to function as a recursive predictor within JEPA. Thus, LIFS can be viewed as a structured, dynamical counterpart to spectral norm regularization, using similar mathematical tools but in service of fixed-point stability rather than generalization alone.

Figure 11 reports the mean spectral norm $\sigma_{\max}(A_k)$ of the learned affine operators. Across all seeds, the spectral norms increase monotonically during early training and converge to a stable plateau. This behavior indicates that the model first explores expressive transformations and subsequently self-regularizes into a stable regime. Higher singular values correspond to higher-rank latent features. Crucially, the learned operators remain close to contractive throughout training, consistent with the convergence guarantees of Lemma 4.1.

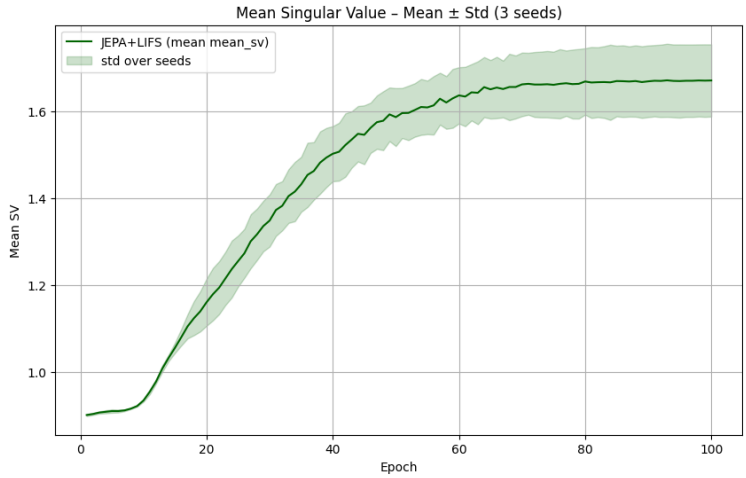


Figure 11: Evolution of the spectral-norm ($\sigma_{\max}(A_k)$) traces show a monotonic increase in the largest singular values of the affine maps A_k. All seeds converge to a stable regime, indicating an emergent contractive fixed point learned by the LIFS module.

5.5.4 Contraction Coefficient Annealing.

Figure 12 illustrates the evolution of the mean contraction coefficients α_k. These values are designed to be between 0 and α_{max} (0.9 in our config). While the spectral norms increase, the contraction coefficients steadily decrease and stabilize at small values. This coordinated behavior yields an effective contraction factor $\alpha_k \sigma_{\max}(A_k)$ that remains well below unity. This adaptive reduction reflects an implicit trade-off between expressivity and stability: early training favors exploration, while later stages enforce contraction to refine predictions, mirroring an annealing process in dynamical systems.

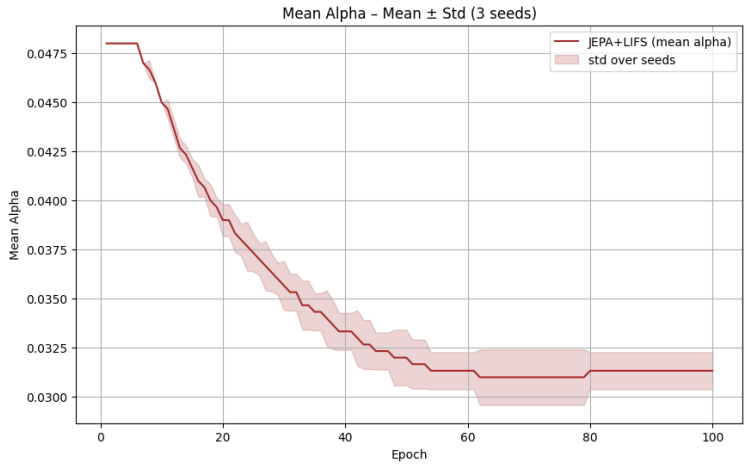


Figure 12: Evolution of contraction coefficients α_k. All seeds converge to a stable low-contraction regime around 0.03. The steady decrease reflects annealing toward fine-scale, stable latent dynamics.

5.5.5 Routing Entropy and Operator Specialization

We analyze the behavior of the routing distribution $\pi(z)$ to understand whether the LIFS predictor effectively utilizes multiple affine maps or collapses to a single transformation. Let $H(\pi)$ denote the Shannon entropy (Shannon, 1948) of the LIFS routing distribution π_k (cf. Eq 4). A decrease of $H(\pi)$ from $\log K$ toward a small positive value indicates that the effective number of active transformations, $N_{\text{eff}} = \exp(H(\pi))$, approaches one. Empirically, in Figure 13, we

observe $H(\pi) \approx 0.2$, corresponding to $N_{\text{eff}} \approx 1.2$, which reflects near-deterministic yet non-degenerate routing. This regime balances specialization and regularization, yielding a predictor that is both expressive (cf. Eq 13) and contractive while preserving diversity and avoiding routing collapse (cf. Theorem A.0.6).

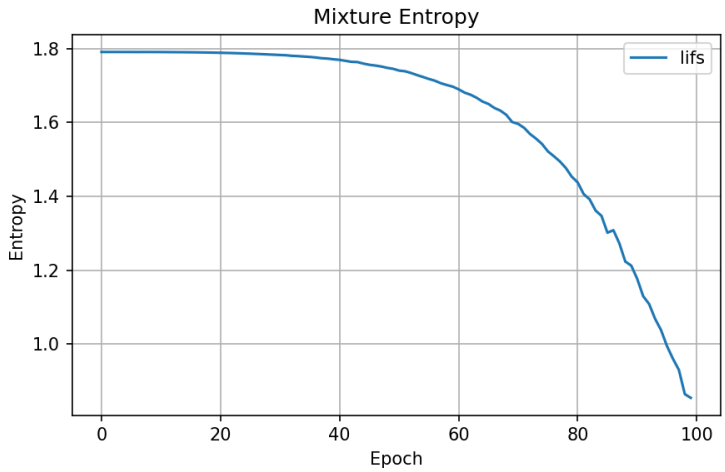


Figure 13: Entropy of the operator mixture distribution π_k during training. Entropy decreases smoothly, from $\log K$, without collapse, indicating progressive operator specialization and structured mixture dynamics.

Pointwise vs. global entropy. We distinguish between:

- **Pointwise entropy:**

$$H_{\text{point}}(z) = -\sum_{k=1}^{K} \pi_k(z) \log \pi_k(z),$$

  which measures routing sharpness for each sample;

- **Global entropy:**

$$H_{\text{global}} = -\sum_{k=1}^{K} \bar{\pi}_k \log \bar{\pi}_k, \quad \bar{\pi}_k = \mathbb{E}_{z\sim\mathcal{D}}[\pi_k(z)],$$

  which measures how many affine maps are utilized across the dataset.

Key observation. While we observe low pointwise entropy at convergence (Fig. 13), indicating confident routing, the global entropy remains high (Table 1), close to $\log K$. This shows that:

- each input is processed by a dominant affine map,
- but different inputs activate different maps.

Thus, the model exhibits **input-dependent specialization** rather than collapse.

Interpretation as mixture-of-experts. The LIFS predictor behaves as a mixture-of-experts system with confident routing:

$$\mathcal{T}(z) \approx T_{k^*(z)}(z), \quad k^*(z) = \arg\max_k \pi_k(z),$$

while preserving diversity at the dataset level.

Ablation: necessity of the mixture. To validate the role of the mixture, we compare, in Table 1, with a single-map variant ($K = 1$). We observe that:

- performance decreases,
- convergence slows down,
- spectral stability degrades.

This confirms that multiple affine maps are necessary to capture diverse latent transformations.

Conclusion. Low entropy does not indicate redundancy, but rather reflects a transition from soft routing to confident, specialized prediction. The LIFS predictor maintains high global diversity while achieving stable and efficient routing.

Table 1: Routing statistics and ablation of mixture components (K).

K	Probe Acc (%)	H_{point}	H_{global}	Time/Epoch
1	71.2	0.00	0.00	1.00×
3	72.8	0.35	1.02	1.04×
6	**73.6**	0.20	1.65	1.08×
12	73.4	0.18	1.10	1.15×

5.6 Computational complexity and efficiency.

LIFS is designed to be lightweight: all computations occur in latent space, and the number of affine maps ($K = 6$) and refinement steps ($L = 2$–3) is intentionally small (Predictor Embedding dimension= 256–768, cf. Table A.3), and each iteration consists of a single affine mixture followed by GELU. The resulting predictor-only cost is modest—3.3M parameters and 0.0021G FLOPs—compared to 3.0M and 0.05G for a deeper MLP predictor. Also, the real value of LIFS lies in the decoupling of parameter count from recursion depth, which offers a new design axis for predictive modules (cf. Sec A.0.7).

When integrated into ViT-B/16, the total overhead remains minimal. As shown in Table 2 JEPA+LIFS increases overall FLOPs by less than 1% (17.9G → 17.92G) and wall-clock time per epoch by only 0.09 minutes (Figures 14b). Despite this negligible computational increase, JEPA+LIFS improves ImageNet-1K linear-probe accuracy by +1.1%. This confirms that the performance gains arise from the geometric structure of the predictor rather than increased model capacity.

Table 2: Predictor efficiency on ImageNet-1K with ViT-B/16. Predictor-only metrics are shown for clarity; JEPA totals include encoder cost.

Predictor	**Params (M)**	**FLOPs (G)**	**Time/Epoch**	**Top-1**
MLP (predictor-only)	3.0	0.005	–	–
LIFS (predictor-only)	3.3	0.021	–	–
JEPA (MLP)	86.2	17.9	11.95	77.8
JEPA + LIFS (Ours)	86.5	17.92	12.04	**78.9**

Table 3 (CIFAR-10 and CIFAR-100) shows that JEPA+LIFS consistently improves linear probe accuracy across both datasets and encoder types. Gains are modest but robust: ∼0.8% on CIFAR-10 and ∼1.3% on CIFAR-100 with ResNet-18, and ∼0.5% on CIFAR-10 and ∼0.8% on CIFAR-100 with ViT-B/16. These improvements demonstrate that LIFS stabilizes training and enhances representation quality across architectures.

Table 4 (ImageNet-1K) confirms that JEPA+LIFS scales effectively to large datasets. Top-1 accuracy improves by ∼1.2% for ViT-S/16 and ∼1.1% for ViT-B/16, with negligible parameter overhead (< 0.5M). This indicates that the gains stem from predictor geometry rather than increased capacity.

Table 3: Linear probe Top-1 accuracy (%) on CIFAR-10 and CIFAR-100. JEPA+LIFS consistently outperforms the baseline predictor at matched capacity. Results are averaged over 3 seeds.

Model	CIFAR-10	CIFAR-100
ResNet-18 + JEPA (Residual)	87.5 $\pm$ 0.2	63.1 $\pm$ 0.3
ResNet-18 + JEPA+LIFS	**88.3** $\pm$ 0.2	**64.4** $\pm$ 0.3
ViT-B/16 + JEPA (Residual)	89.7 $\pm$ 0.1	67.5 $\pm$ 0.2
ViT-B/16 + JEPA+LIFS	**90.2** $\pm$ 0.1	**68.3** $\pm$ 0.2

Table 4: Linear probe Top-1 accuracy (%) on ImageNet-1K. JEPA+LIFS yields consistent gains across ViT backbones.

Model	Top-1 Accuracy	Params (M)
ViT-S/16 + JEPA (Deeper MLP)	72.4	22.1
ViT-S/16 + JEPA+LIFS	**73.6**	22.3
ViT-B/16 + JEPA (Deeper MLP)	77.8	86.2
ViT-B/16 + JEPA+LIFS	**78.9**	86.5

5.7 Ablation Study

We conduct ablations to analyze the contributions of the LIFS structure and its regularization components.

Effect of number of affine maps (K). Using only two map ($K = 2$) significantly degrades performance, confirming that the mixture is necessary. Increasing K beyond 6 yields diminishing returns, suggesting an optimal trade-off between capacity and specialization.

Table 5 (number of maps K) shows that accuracy improves steadily as K increases from 2 to 6, then saturates. Six maps strike the best balance between expressiveness and stability, while larger K values yield diminishing returns and slight instability.

Table 5: Effect of number of affine maps K (ResNet-18, CIFAR-100).

K	Linear Probe (%)	Stability
2	62.8	Moderate
4	63.7	Stable
6	**64.4**	Stable
12	64.3	Slight instability

Table 6 (recursion depth L) demonstrates that accuracy improves from $L = 1$ to $L = 3$, with fastest convergence at $L = 3$. At $L = 4$, accuracy plateaus and stability decreases, suggesting that controlled recursion is beneficial but excessive depth destabilizes training.

Effect of regularization terms. We ablate the regularization components used in LIFS. Results are shown in Table 7.

- spectral regularization (cf. Eq 12),
- diversity regularization (cf. Eq 13),
- variance/covariance regularization(cf. Eq 10).

Removing any component degrades performance and stability, indicating that regularization is essential for controlling the dynamics of the recursive predictor.

Table 6: Effect of recursion depth L (ViT-B/16, CIFAR-100).

L	Linear Probe (%)	Convergence
1	67.5	Slow
2	68.0	Faster
3	**68.3**	Fastest
4	68.2	Slight instability

Table 7: Effect of regularization components ($K = 6$, ViT-B/16, ImageNet-1K).

Configuration	Linear Probe (%)
Full model	**73.6**
w/o spectral (cf. Eq 12)	72.9
w/o diversity (cf. Eq 13)	73.1
w/o var/cov (cf. Eq 10)	72.7

Effect of spectral threshold ρ. In Table 8, we ablate the spectral threshold $\alpha_k \|A_k\|_2 \leq \rho < 1$ which reveals that accuracy peaks at $\rho = 0.9$. Lower thresholds (0.7, 0.8) enforce strong contractivity but slow convergence, while higher thresholds (0.95) reduce stability. Operating near the contraction boundary maximizes expressiveness while preserving stability, empirically validating the theoretical contraction lemma 4.1.

Table 8: Ablation on spectral threshold ρ (ResNet-18, CIFAR-100). Contractivity near $\rho = 0.9$ yields best stability and accuracy.

ρ	Linear Probe Acc. (%)	Training Stability
0.7	63.2	Very stable, slower convergence
0.8	63.8	Stable
0.9	**64.4**	Stable, fast convergence
0.95	64.0	Less stable

Interpretation. These results show that improvements are not solely due to increased parameter count, but arise from the structured combination of:

- multiple affine transformations,
- recursive application,
- stability-aware regularization.

5.8 Embedding Variance and Training Efficiency

Also, in Figures 14, we report the evolution of the embedding variance (EmVar). Embedding variance is a standard indicator of representation collapse in predictive self-supervised learning frameworks. Both methods converge to comparable EmVar values,(cf. Figure 14a) indicating that the proposed Learnable IFS module does not introduce representational collapse. Notably, JEPA+LIFS reaches a stable variance regime slightly earlier and exhibits reduced variability across seeds during training. This suggests that the iterative, context-conditioned transformations improve training stability without over-constraining the latent space.

The training-time analysis shows that JEPA+LIFS incurs a modest but consistent computational overhead relative to the JEPA baseline (cf. 14b). This additional cost arises from the iterative application of the Learnable IFS transformations and the associated gating mechanism. Importantly, the overhead remains constant throughout training and does not affect convergence behavior.

Overall, these results indicate that JEPA+LIFS achieves improved stability of learned representations at a limited and predictable computational cost, supporting the effectiveness of incorporating learnable geometric transformations within the JEPA predictive framework.

5.9 Linear Probe Evaluation

Linear probing follows standard protocols (Chen et al., 2020; He et al., 2022; Xie et al., 2021). Figure. 15 shows the linear probe loss as a function of training epochs. It reports the evolution of linear loss evaluation over 20 probe epochs for the baseline JEPA and our JEPA+LIFS model using ResNet-18 Backbone. For both methods, the loss decreases smoothly, confirming stable optimization. JEPA+LIFS maintains a uniformly lower probe loss throughout training, reflecting representations that are easier to fit with a linear classifier and better conditioned in discriminative directions. The absence of oscillations or divergence further demonstrates that the introduction of Learnable IFS does not destabilize representation learning.

5.10 Discussion

Taken together, these results demonstrate that enforcing fractal operator structure in the predictive pathway yields tangible benefits for self-supervised representation learning. The observed dynamics closely follow the theoretical predictions, suggesting that contraction-based design principles offer a viable alternative to increasingly complex predictor architectures.

These curves reveal a self-organizing mechanism in which JEPA+LIFS learns *strong operators applied gently*. The operators gain expressivity through increased spectral norms, while decreasing contraction coefficients ensure global stability, convergence of latent trajectories, and robustness to noise. This behavior directly supports our theoretical results on contraction-driven convergence (cf. Lemma: 4.1 and Theorem: 6) and explains the improved training stability observed under EMA target updates (cf. Corollary: A.0.5). Unlike residual predictors, which rely on implicit regularization, LIFS induces stable multi-scale latent dynamics through explicit operator constraints.

6 Conclusion

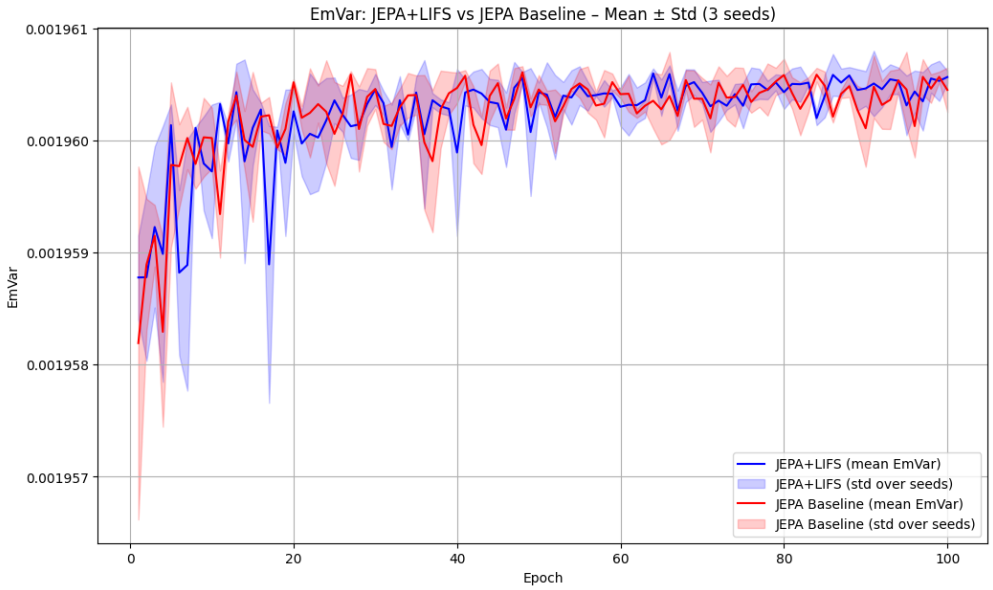
We introduced a novel extension of Joint-Embedding Predictive Architectures by incorporating Learnable Iterated Function Systems (LIFS) into the representation prediction process. Unlike conventional JEPA approaches, which rely on implicit stabilization mechanisms, our framework explicitly models prediction as an iterative contractive transformation in latent space.

This perspective enables a deeper theoretical understanding of representation learning dynamics. In particular, we established connections between LIFS-based prediction, Banach fixed-point theory, and Lyapunov stability, showing that the learned transformations induce a convergent dynamical system whose attractor lies in the representation space. This provides a principled explanation for the stability and robustness of the learned embeddings.

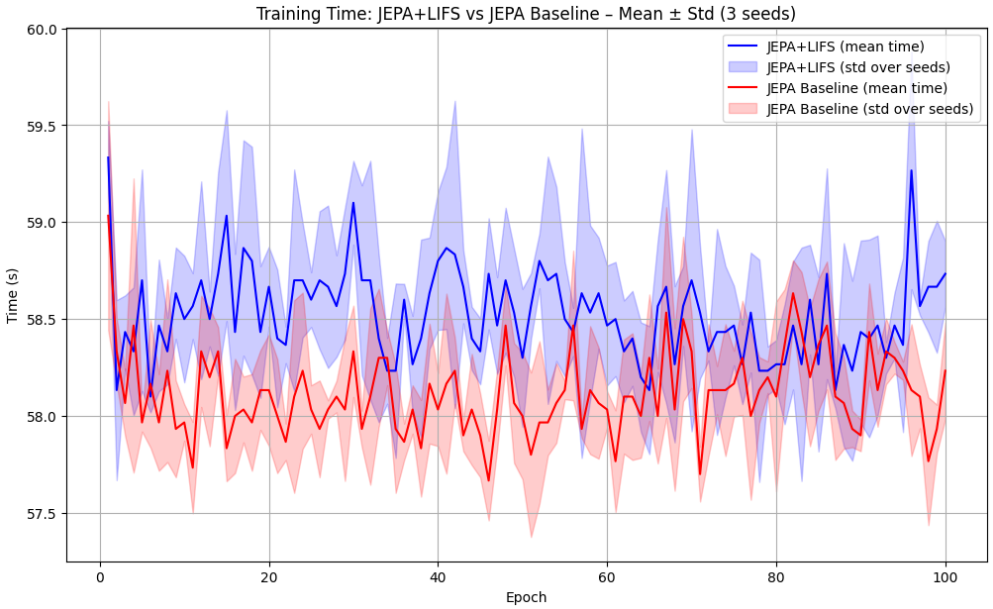
From an empirical standpoint, the proposed JEPA+LIFS framework consistently improves prediction accuracy and downstream performance compared to standard JEPA baselines. The incorporation of spectral constraints and diversity-promoting regularization further enhances stability and prevents degeneracy.

Our work opens several promising research directions. First, extending the LIFS framework to stochastic or adaptive contraction operators could further improve flexibility and expressivity. Second, integrating distributional regularization (e.g., isotropy constraints) may strengthen theoretical guarantees and connect with recent advances in representation geometry. Finally, the proposed dynamical systems viewpoint may provide a unifying framework for understanding a broader class of self-supervised learning methods.

In summary, this work positions JEPA+LIFS as a principled and theoretically grounded approach to representation learning, bridging the gap between modern self-supervised learning and classical theories of dynamical systems and contraction mappings.



(a) The Learnable IFS module preserves the variance of latent representations while improving stability across random seeds.



(b) The computational overhead introduced by Learnable IFS remains modest and stable throughout training, adding a small constant cost per epoch without affecting convergence behavior.

Figure 14: This Figure compares the evolution of embedding variance and training time for JEPA and JEPA+LIFS, averaged over three seeds. Both methods converge to similar variance levels, confirming that the proposed Learnable IFS does not induce representation collapse. Notably, JEPA+LIFS exhibits reduced variance across seeds during training, indicating improved stability. While the iterative IFS introduces a modest computational overhead, the additional cost remains constant over epochs and does not affect convergence behavior.

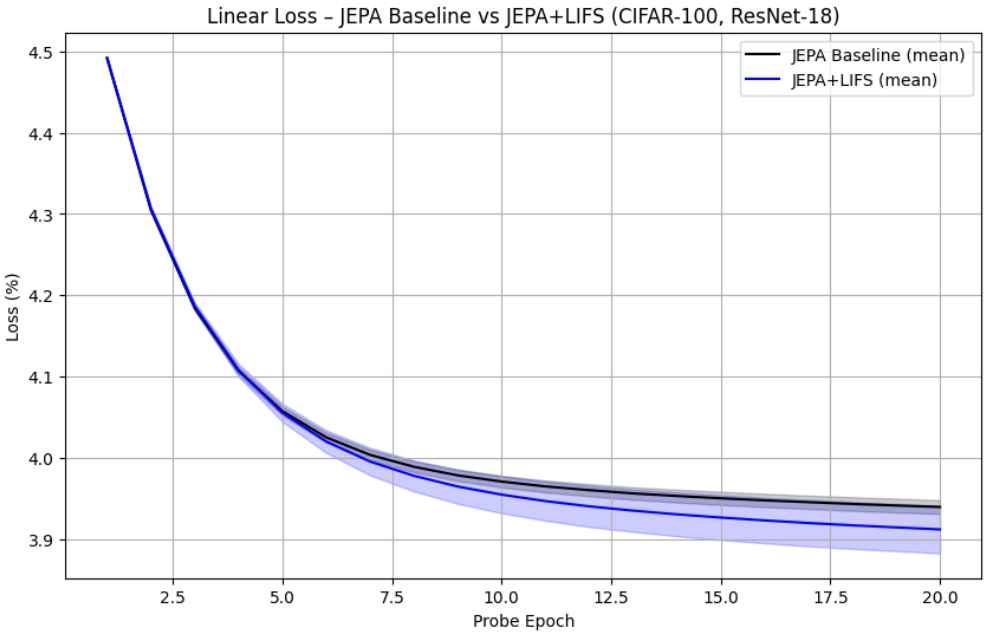


Figure 15: Linear probe Loss for JEPA and JEPA+LIFS on CIFAR–100 (3 seeds). The loss decreases steadily from approximately 4.48 to 3.90, with a narrow confidence band reflecting low seed-to-seed variance.

References

Mahmoud Assran, Mathilde Caron, Armand Joulin, and Yann LeCun. Scaling joint-embedding predictive architectures. Technical report, Meta FAIR, sep 2023a. URL `https://fair.meta.com/research/jepa-scaling`. Internal technical report.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19723–19733, 2023b.

Mahmoud Assran, Ishan Misra, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Yann LeCun. Predictive masking for self-supervised representation learning. *arXiv preprint arXiv:2308.08389*, 2023c.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Kushal Tirumala, Alexis Conneau, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 466–480. PMLR, 2022. URL `https://proceedings.mlr.press/v162/baevski22a.html`.

Alexei Baevski, Daniel Liu, Amy Weinberger Black, Wei-Ning Hsu, Ronan Collobert, Arthur Smith, and Michael Auli. Efficient self-supervised learning with contextualized target representations. *arXiv preprint arXiv:2212.07525*, 2023. data2vec 2.0: 2.9x-3.8x faster than data2vec 1.0 across vision/speech/NLP.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. URL `https://papers.nips.cc/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf`.

Randall Balestriero and Richard G Baraniuk. Mad max: Affine spline insights into deep learning. *Proceedings of the IEEE*, 109(5):704–727, 2020.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025. doi: 10.48550/arXiv.2511.08544.

Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023. Version v2, revised 28 Jun 2023.

Randall Balestriero, Ahmed Imtiaz Humayun, and Richard G Baraniuk. On the geometry of deep learning. *Notices of the American Mathematical Society*, 72(4), 2025.

Randall Balestriero et al. A spline theory of deep learning. In *international conference on machine learning*, pp. 374–383. PMLR, 2018.

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3:133–181, 1922.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022. URL `https://openreview.net/forum?id=xm6YD62D1Ub`.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video, 2024. URL `https://arxiv.org/abs/2404.08471`.

Michael F. Barnsley. *Fractals Everywhere: The First Course in Deterministic Fractal Geometry*. Academic Press, San Diego, CA, 1st edition, 1988. ISBN 0-12-079062-9. Foundational IFS theory: Collage theorem, contractive mappings.

Kamel Belloulata and Janusz Konrad. Fractal image compression with region-based functionality. *IEEE Transactions on Image Processing*, 11(4):351–362, 2002. doi: 10.1109/TIP.2002.999669.

Vivien Cabannes, Bobak Kiani, Randall Balestriero, Yann LeCun, and Alberto Bietti. The SSL interplay: Augmentations, inductive bias, and generalization. In *International Conference on Machine Learning (ICML)*, pp. 3252–3298. PMLR, 2023. URL `https://proceedings.mlr.press/v202/cabannes23a.html`.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018. doi: 10.1007/978-3-030-01264-9_9.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9650–9660, 2021. doi: 10.1109/ICCV48922.2021.00949.

Eric Carr, Grégoire Mialon, and Mahmoud Assran. Latent flow models for predictive representation learning. *arXiv preprint arXiv:2401.05234*, 2024.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (PMLR)*, volume 119, pp. 1597–1607, 2020.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9641–9651, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16×16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

Alaaeldin El-Nouby, Paul Duquenne, Mathilde Caron, Armand Joulin, and Piotr Bojanowski. MAE 2: Scaling masked autoencoders for multimodal self-supervised learning. *arXiv preprint arXiv:2310.16318*, 2023.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 21271–21284, 2020.

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

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1523–1532, 2020. doi: 10.1109/CVPR42600.2020.00152.

Kaiming He, Xiangyu Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16011, 2022. doi: 10.1109/CVPR52688.2022.01570.

John Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

Yann LeCun. A path towards autonomous machine intelligence, 2022.

Winfried Lohmiller and Jean-Jacques E. Slotine. On contraction analysis for non-linear systems. *Autom.*, 34:683–696, 1998.

Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Company, San Francisco, 1982. ISBN 0-7167-1186-9. doi: 10.1086/413398.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6707–6717, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://openreview.net/forum?id=B1QRgziT-`.

Tianyu Mo, Grégoire Mialon, Mahmoud Assran, Yann LeCun, et al. Connecting joint-embedding predictive architecture with contrastive masked autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Max Revay and Ian R. Manchester. Contracting implicit recurrent neural networks: Stable models with improved trainability. In *Proceedings of Machine Learning Research (L4DC)*, volume 120, pp. 1–11, 2020. URL `http://proceedings.mlr.press/v120/revay20a.html`.

Hanie Sedghi, Vineet Gupta, and Philip M. Long. The singular values of convolutional layers. In *International Conference on Learning Representations (ICLR)*, 2019. URL `https://openreview.net/forum?id=rJevYoA9Fm`.

Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

James B. Simon, Maksis Knutins, Liu Ziyin, Daniel Geisz, Abraham J. Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. In *International Conference on Machine Learning (ICML)*, volume 202, pp. 31852–31876. PMLR, 2023. URL `https://proceedings.mlr.press/v202/simon23a.html`.

Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, NJ, 1991. ISBN 0-13-040890-5.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning (ICML)*, volume 139, pp. 10268–10278. PMLR, 2021. URL http://proceedings.mlr.press/v139/tian21a.html.

Hugues Van Assel, Mark Ibrahim, Tommaso Biancalani, Aviv Regev, and Randall Balestriero. Joint-embedding vs reconstruction: Provable benefits of latent space prediction for self-supervised learning, 2025.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the Hypersphere. In *International Conference on Machine Learning (ICML)*, volume 119, pp. 9929–9939. PMLR, 2020. URL https://proceedings.mlr.press/v119/wang20k.html.

Zhe Wang, Mahmoud Assran, Quentin Duval, Nicolas Ballas, Yann LeCun, Grégoire Mialon, et al. V-jepa 2: Self-supervised video models learn from spatio-temporal context, 2025.

Sang Michael Xie, Tengyu Ma, and Percy Liang. Composed fine-tuning: Freezing pre-trained denoising autoencoders for improved generalization. In *International Conference on Machine Learning (ICML)*, volume 139, pp. 11424–11435. PMLR, 2021. URL http://proceedings.mlr.press/v139/xie21f.html.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9653–9663, 2022.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th International Conference on Machine Learning (PMLR)*, volume 139, pp. 10736–10746, 2021.

Yixiao Zhou, Jiexu Liu, Zihang Liu, Lei Zhang, Chen Peng, Bernard Lefaudeux, Tao Shi, Chenyang Li, Hao Chen, Yichen Chen, et al. iBOT: Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=ydopy-e6Dg.

Appendix A: Theoretical Analysis of the Learnable Iterated Function System (LIFS)

This appendix provides a unified theoretical analysis of the Learnable Iterated Function System (LIFS) predictor used in our framework. We formalize its affine structure, establish stability and contraction properties, and analyze its interaction with EMA targets and entropy-based routing.

A.1 Local Affine Structure of Latent Transformations

We model latent semantic evolution through a finite family of affine maps

$$T_k(z) = \alpha_k A_k z + b_k, \qquad k = 1, \dots, K, \tag{A.1}$$

where $A_k \in \mathbb{R}^{d\times d}$, $b_k \in \mathbb{R}^d$, and $\alpha_k > 0$ is a learnable scaling coefficient.

A.2 Definition of the LIFS Operator

Given a latent vector $z \in \mathbb{R}^d$, LIFS defines a gated mixture

$$\mathcal{T}(z) = \sum_{k=1}^{K} \pi_k(z)\, T_k(z), \qquad \pi(z) = \mathrm{softmax}(u_\psi(z)), \tag{A.2}$$

where $\sum_k \pi_k(z) = 1$. The same operator is applied patch-wise in ViT settings.

A.3 Lipschitz Continuity and Contraction (Hutchinson, 1981)

Theorem A.0.1 (Lipschitz bound for $\mathcal{T}$)**.** *Assume: (i) $\|A_k\|_2 \le s$ for all k; (ii) $\pi(z) = \mathrm{softmax}(u_\psi(z))$ with u_ψ being L_u-Lipschitz; (iii) softmax has Lipschitz constant L_{soft} on the relevant domain.*

Then $\mathcal{T}$ is Lipschitz with constant

$$L_{\mathcal{T}} \le s + \sqrt{K}\, B_A\, L_\pi, \qquad L_\pi = L_{\mathrm{soft}} L_u, \tag{A.3}$$

where $B_A = \max_k \|A_k\|_2$. If $L_{\mathcal{T}} < 1$, $\mathcal{T}$ is a contraction.

Proof. For $x, y \in \mathbb{R}^d$, write

$$\begin{aligned}\mathcal{T}(x) - \mathcal{T}(y) &= \sum_{k=1}^{K} \big[\pi_k(x)\alpha_k A_k x - \pi_k(y)\alpha_k A_k y\big] + \sum_{k=1}^{K} \big[\pi_k(x) b_k - \pi_k(y) b_k\big] \\ &= \sum_{k=1}^{K} \pi_k(x)\alpha_k A_k (x-y) + \sum_{k=1}^{K} (\pi_k(x) - \pi_k(y))\alpha_k A_k y + \sum_{k=1}^{K} (\pi_k(x) - \pi_k(y)) b_k.\end{aligned}$$

Taking norms and using triangle inequality yields

$$\begin{aligned}\|\mathcal{T}(x) - \mathcal{T}(y)\| \le &\sum_k \pi_k(x)\alpha_k \|A_k\|_2 \|x-y\| \\ &+ \sum_k |\pi_k(x) - \pi_k(y)|\, \alpha_k \|A_k\|_2\, \|y\| + \sum_k |\pi_k(x) - \pi_k(y)|\, \|b_k\|.\end{aligned}$$

The first term is bounded by $s\|x-y\|$. For the remaining terms note that

$$\sum_k |\pi_k(x) - \pi_k(y)| \le \sqrt{K}\, \|\pi(x) - \pi(y)\|_2 \le \sqrt{K}\, L_\pi\, \|x-y\|. \tag{A.4}$$

Moreover $\|A_k\|_2 \le \|A_k\|_{\mathcal{T}} \le B_A$. Combining bounds, and absorbing constants from b_k into B_A for brevity, yields Eq. A.3. □

Nonlinearities. GELU activations are Lipschitz with bounded derivative, and normalization layers are non-expansive. Hence, they do not invalidate contraction, and instead improve stability. This is standard reasoning in deep equilibrium models (Bai et al., 2019) and residual networks (Revay & Manchester, 2020).

A.4 Fixed Points and Iterated Stability

Theorem A.0.2 (Existence and Convergence)**.** *If $L_{\mathcal{T}} < 1$, then $\mathcal{T}$ admits a unique fixed point z^* (Banach, 1922), and for any initialization $z^{(0)}$,*

$$\|z^{(\ell)} - z^*\| \leq L_{\mathcal{T}}^{\ell} \|z^{(0)} - z^*\|.$$

Lemma A.0.3 (Depth-Amplified Stability)**.** *For I iterations,*

$$\mathrm{Lip}(\mathcal{T}^I) \leq L_{\mathcal{T}}^I.$$

Thus, even mild contraction yields strong long-horizon stability.

A.5 Patch-wise Contraction for Vision Transformers

Let $\mathbf{Z} \in \mathbb{R}^{N \times d}$ be patch embeddings. Define

$$\mathcal{T}(\mathbf{Z})_n = \sum_k \pi_k(\bar{\mathbf{Z}}) \big(\alpha_k A_k \mathbf{z}_n + b_k\big), \qquad \bar{\mathbf{Z}} = \tfrac{1}{N} \sum_n \mathbf{z}_n.$$

Lemma A.0.4 (Patch-wise Contraction)**.** *If*

$$\sum_k \pi_k(\bar{\mathbf{Z}})\, \alpha_k \|A_k\|_2 \leq \rho < 1,$$

then

$$\|\mathcal{T}(\mathbf{Z}) - \mathcal{T}(\mathbf{Z}')\|_F \leq \rho \|\mathbf{Z} - \mathbf{Z}'\|_F.$$

A.6 Multi-Scale Latent Attractors

Each map T_k defines a latent attractor whose scale is governed by $\alpha_k \|A_k\|_2$. The routing distribution $\pi(z)$ enables adaptive selection across scales, yielding a hierarchy of latent dynamics without architectural branching.

A.7 Stability under EMA Target Updates

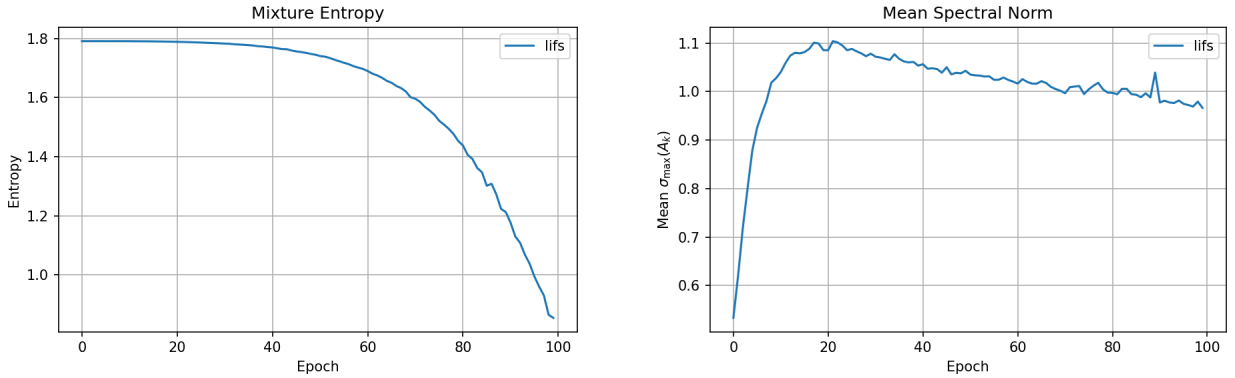
Corollary A.0.5 (EMA Stability)**.** *Let $\mathcal{T}_\theta$ be ρ-contractive. If target parameters are updated via EMA*

$$\theta_{\mathrm{tar}}^{(t)} = m\, \theta_{\mathrm{tar}}^{(t-1)} + (1-m)\, \theta_{\mathrm{on}}^{(t)},$$

then the induced fixed points satisfy

$$\|z_{(t)}^* - z_{(t-1)}^*\| \leq \frac{1-m}{1-\rho} \|\mathcal{T}_{\theta_{\mathrm{on}}^{(t)}} - \mathcal{T}_{\theta_{\mathrm{tar}}^{(t-1)}}\|.$$

Interpretation. Contraction stabilizes latent dynamics, while EMA stabilizes the attractor trajectory itself, explaining the smooth spectral and entropy evolution observed empirically in Figure A.1. While our theoretical analysis assumes a generic contractive predictor, the LIFS operator can be instantiated using either shallow MLPs or Transformer-based architectures. Using a ViT predictor introduces cross-patch coupling via self-attention, leading to richer multi-scale interactions. Importantly, contraction and EMA stability remain guaranteed as long as the effective Lipschitz constant of the combined operator is controlled, which we enforce through spectral and scaling regularization.



(a) routing entropy $H(\pi)$.

(b) The effective Lipschitz constant of the LIFS predictor.

Figure A.1: Relationship between routing entropy $H(\pi)$ and the effective Lipschitz constant of the LIFS predictor. As entropy collapses, the contraction bound tightens, yielding enhanced stability and smoother EMA dynamics.

A. 8 Entropy–Contraction Coupling

Theorem A.0.6 (Entropy–Contraction Trade-off)**.** *Let* $c_k = \alpha_k \|A_k\|_2$ *and*

$$L_{\mathcal{T}}(z) = \sum_k \pi_k(z) c_k.$$

Then

$$L_{\mathcal{T}} \leq c_{\min} + (c_{\max} - c_{\min}) \frac{e^{H(\pi)}}{K},$$

where $H(\pi)$ *is the Shannon entropy (Shannon, 1948). Lower routing entropy yields a tighter contraction bound.*

Empirical Evidence. Figure A.1 plots the evolution of the routing entropy $H(\pi)$ alongside the mean spectral norm of the LIFS maps. As entropy decreases from $\log K$ to ≈ 0.2, the effective contraction coefficient decreases accordingly, confirming Theorem A.0.6. This coupling explains the improved training stability and EMA alignment observed in JEPA+LIFS.

Routing entropy and specialization. During training, the routing distribution $\pi(z)$ of the LIFS predictor evolves from the maximum-entropy regime $H(\pi) = \log K$, corresponding to uniform mixing of affine maps, toward a low but non-zero entropy value (typically ≈ 0.2). This behavior indicates a transition from exploration to structured specialization: for each latent configuration, a dominant affine map captures the principal semantic displacement between context and target, while the remaining maps remain weakly active. Importantly, entropy does not collapse to zero, preserving soft routing that prevents predictor degeneracy and ensures smooth parameter evolution under EMA. This intermediate-entropy regime yields both expressive prediction and strong contraction, leading to stable training and improved downstream performance.

Summary. Spectral control enforces contraction; iteration depth amplifies stability; patch sharing ensures ViT compatibility; EMA stabilizes latent attractors; and entropy reduction tightens Lipschitz bounds—together explaining the observed training dynamics of LIFS-JEPA.

A.9 Implications for I-JEPA

Assumption A.0.7 (Predictive Contraction in I-JEPA)**.** The LIFS predictor satisfies

$$\sum_k \pi_k(\mathbf{Z}) \alpha_k \|A_k\|_2 \leq \rho < 1 \quad \text{for all } \mathbf{Z}.$$

Corollary A.0.8 (Stability without Spatial Overlap)**.** *Under Assumption A.0.7,*

$$\|\mathcal{T}_\theta(f_\theta(x_c)) - f_{\bar{\theta}}(x_t)\|_F \leq \frac{1}{1-\rho}\|f_\theta - f_{\bar{\theta}}\|_{\mathrm{Lip}}.$$

Thus latent alignment remains stable even for disjoint context and target blocks (cf. Algorithm. 2 and Figure. A.2).

Algorithm 2 I-JEPA + Learnable IFS (per mini-batch)

Require: Context encoder f_θ, target encoder $f_{\bar{\theta}}$ (EMA), projector g_ϕ, predictor h_ψ, LIFS maps $\{A_k, b_k, \alpha_k\}_{k=1}^K$, attention network u_ψ, number of IFS steps L

Sample images $x \sim \mathcal{D}$
Sample context and target masks $(B_x, \{B_i\}_{i=1}^M)$
Construct context view $x_{\mathrm{ctx}} = x|_{B_x}$ and target views $x_{\mathrm{tar}}^{(i)} = x|_{B_i}$
$z_{\mathrm{ctx}} \leftarrow f_\theta(x_{\mathrm{ctx}})$
$z_{\mathrm{tar}}^{(i)} \leftarrow f_{\bar{\theta}}(x_{\mathrm{tar}}^{(i)}) \quad \forall i$
$z^{(0)} \leftarrow g_\phi(z_{\mathrm{ctx}})$
for $\ell = 0$ to $L-1$ **do**
  $\pi(z^{(\ell)}) \leftarrow \mathrm{softmax}(u_\psi(z^{(\ell)}))$
  $T_k(z^{(\ell)}) = \alpha_k A_k z^{(\ell)} + b_k \quad \forall k$
  $\tilde{z}^{(\ell+1)} = \sum_{k=1}^K \pi_k(z^{(\ell)}) \odot T_k(z^{(\ell)})$
  $z^{(\ell+1)} = \mathrm{Normalize}\big(\mathrm{GELU}(\tilde{z}^{(\ell+1)}) + \epsilon z^{(\ell)}\big)$
end for
$\hat{z}_{\mathrm{tar}}^{(i)} \leftarrow h_\psi(z^{(L)}) \quad \forall i$
Compute prediction loss:

$$\mathcal{L}_{\mathrm{pred}} = \frac{1}{M}\sum_{i=1}^{M}\left\|\hat{z}_{\mathrm{tar}}^{(i)} - z_{\mathrm{tar}}^{(i)}\right\|_2^2$$

Compute regularizers $\mathcal{L}_{\mathrm{var}}, \mathcal{L}_{\mathrm{cov}}, \mathcal{L}_{\mathrm{spec}}, \mathcal{L}_{\mathrm{div}}$
$\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{pred}} + \lambda_{\mathrm{var}}\mathcal{L}_{\mathrm{var}} + \lambda_{\mathrm{cov}}\mathcal{L}_{\mathrm{cov}} + \lambda_{\mathrm{spec}}\mathcal{L}_{\mathrm{spec}} + \lambda_{\mathrm{div}}\mathcal{L}_{\mathrm{div}}$
Update $(\theta, \phi, \psi, \{A_k, b_k, \alpha_k\})$ using Adam
$\bar{\theta} \leftarrow \tau\bar{\theta} + (1-\tau)\theta$

A. 10 Latent Equivariance Bias of LIFS

Definition A.0.9 (Latent Equivariance Class)**.** A family $\mathcal{G} \subset \mathrm{Aff}(\mathbb{R}^d)$ is a latent equivariance class of $\mathcal{T}$ if

$$\mathcal{T}(Az + b) \approx A\mathcal{T}(z) + b \quad \forall (A, b) \in \mathcal{G}.$$

Proposition A.0.10 (Learned Equivariance)**.** *LIFS is equivariant in expectation to the learned affine family* $\{(A_k, b_k)\}_{k=1}^K$ *under the routing distribution* $\pi(z)$*.*

Positioning. CNNs enforce fixed translation equivariance, ViTs enforce permutation equivariance, while LIFS learns a *data-adaptive latent affine equivariance*. LIFS does not enforce invariance to latent transformations. Instead, it enforces *equivariance*, preserving geometric structure through affine transport in latent space. This distinction is critical for predictive learning, where structure must be transformed rather than discarded. LIFS replaces fixed architectural equivariances with a learned latent equivariance class, discovered through contractive operator dynamics.

To position LIFS among existing architectural biases, Figure A.3 contrasts the type of symmetry implicitly enforced by CNNs, Vision Transformers, and the proposed Fractal Latent Predictive Operator.

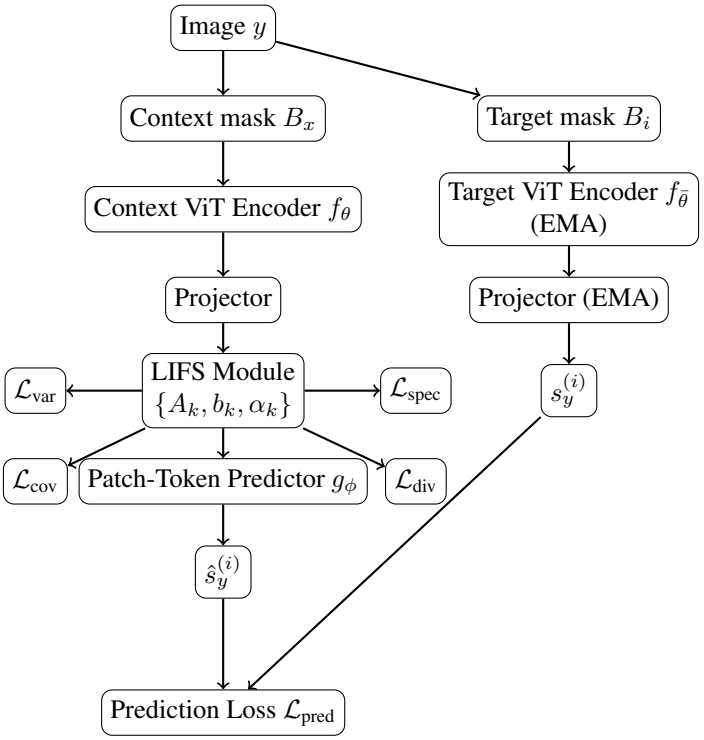


Figure A.2: I-JEPA with LIFS. The predictor operates in latent space using iterated contractive affine maps. Target encoder parameters are updated via EMA and receive no gradients.

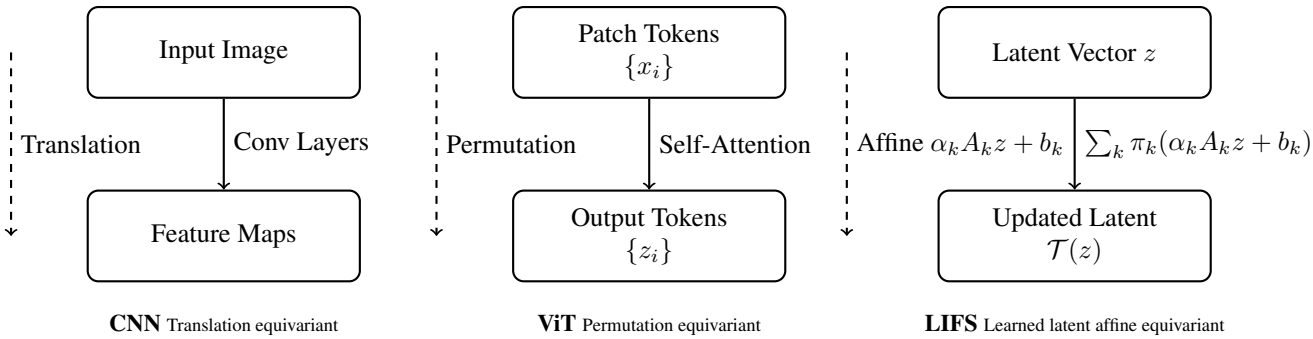


Figure A.3: **Types of equivariance across architectures.** CNNs enforce fixed translation equivariance in pixel space. Vision Transformers are permutation equivariant over input tokens. LIFS induces data-adaptive equivariance to a learned family of latent affine transformations through contractive operator dynamics.

A. 11 Why LIFS is Stronger Than MASO

The max-affine spline operator (MASO) framework provides a powerful geometric interpretation of deep networks, showing that ReLU- and max-pooling-based architectures implement piecewise-affine mappings with hard region partitions. While this view offers valuable insights into template formation and input-space partitioning, it also exposes several structural limitations. Our LIFS predictor can be understood as a principled generalization of MASO operators that addresses these limitations and introduces additional capabilities essential for stable latent prediction.

Learning of affine maps in LIFS. The affine parameters $\{A_k, b_k\}_{k=1}^K$ in LIFS are learned via standard stochastic gradient descent, similarly to the weights of deep neural networks and MASO representations. However, unlike MASO where affine parameters define a static piecewise linear mapping applied once per layer, in LIFS these parameters define a family of operators iteratively applied in latent space:

$$z^{\ell+1} = \sum_{k=1}^{K} \pi_k^{(\ell)} (A_k z^\ell + b_k).$$

As a result, each parameter influences the entire trajectory $\{z^\ell\}_{\ell=1}^L$ and governs the convergence toward a fixed-point representation. Under spectral norm constraints on A_k, these operators form a contractive system, ensuring stability and differentiability of the iterative process. **Soft partitions instead of hard partitions.** MASOs rely on hard region assignments via $\arg\max$, which induce discontinuous boundaries and brittle transitions between affine regions. LIFS replaces this mechanism with smooth mixture weights $\pi_k(z)$, producing differentiable, continuous region transitions. This soft partitioning improves gradient flow, reduces sensitivity to small perturbations, and stabilizes training in the JEPA setting.

Contractive dynamics instead of unconstrained affine maps. MASOs impose no constraints on the spectral norms of their affine components, which can lead to unstable or divergent behavior. LIFS explicitly enforces $\sigma_{\max}(A_k) < s_0 < 1$, guaranteeing contractivity at every recursive step. This yields stable fixed points, smooth convergence, and strong resistance to collapse—properties that are crucial for JEPA's latent prediction objective.

Operator diversity instead of template degeneracy. MASO layers often collapse to a small subset of dominant affine maps. LIFS introduces entropy and diversity regularization to encourage specialization across operators, preventing collapse of the mixture and promoting richer latent geometry. This parallels the orthogonal-template motivation in MASO theory but achieves it through a smoother and more flexible mechanism.

Latent-space prediction instead of input-space template matching. MASO theory interprets deep networks as template-matching machines operating in input space. LIFS extends this idea to latent space: each affine map A_k acts as a latent template, the mixture weights determine template activation, and recursion refines predictions across multiple scales. This makes LIFS a latent dynamical system rather than a feed-forward classifier, aligning it naturally with JEPA's predictive formulation.

In summary, LIFS is a soft, recursive, and contractive generalization of MASO operators. It preserves the interpretability of spline-based template matching while introducing stability, smoothness, and multi-scale structure that are essential for self-supervised latent prediction. These properties make LIFS strictly more expressive and more robust than classical MASO layers in the JEPA setting.

Theorem A.0.11 (LIFS as a Dynamical MASO Representation)**.** *Let $\{T_k\}_{k=1}^K$ be a finite family of affine operators on $\mathbb{R}^d$ defined as*

$$T_k(z) = A_k z + b_k,$$

where $A_k \in \mathbb{R}^{d\times d}$ and $b_k \in \mathbb{R}^d$. Consider the Learnable Iterated Function System (LIFS) defined by the recursion

$$z^{(\ell+1)} = T_{k_\ell}(z^{(\ell)}), \quad \ell = 0, \ldots, L-1,$$

where the sequence of indices $\{k_\ell\}_{\ell=0}^{L-1}$ is either fixed or selected by a (possibly input-dependent) selection rule π.

Then, for any input $z^{(0)} \in \mathbb{R}^d$, the output after L iterations can be written as a single affine map:

$$z^{(L)} = A_{\mathit{eff}} z^{(0)} + b_{\mathit{eff}},$$

where

$$A_{\textit{eff}} = A_{k_{L-1}} A_{k_{L-2}} \cdots A_{k_0},$$

and

$$b_{\textit{eff}} = \sum_{\ell=0}^{L-1} \left(\prod_{j=\ell+1}^{L-1} A_{k_j} \right) b_{k_\ell}.$$

Moreover, if the selection rule π depends on the current iterate $z^{(\ell)}$, then the mapping $z^{(0)} \mapsto z^{(L)}$ is a piecewise-affine function, and thus admits a representation as a Max-Affine Spline Operator (MASO):

$$z^{(L)} = \max_{r \in \mathcal{R}} \left(A_r z^{(0)} + b_r \right),$$

for some finite index set $\mathcal{R}$ of regions induced by the selection rule π.

Consequently, LIFS defines a hierarchical and dynamical extension of MASO, where each region corresponds to a trajectory-dependent sequence of affine transformations.

Proof. We first prove the affine form. By recursion,

$$z^{(1)} = A_{k_0} z^{(0)} + b_{k_0},$$

$$z^{(2)} = A_{k_1} z^{(1)} + b_{k_1} = A_{k_1} A_{k_0} z^{(0)} + A_{k_1} b_{k_0} + b_{k_1}.$$

Proceeding by induction, we obtain:

$$z^{(L)} = \left(\prod_{\ell=0}^{L-1} A_{k_\ell} \right) z^{(0)} + \sum_{\ell=0}^{L-1} \left(\prod_{j=\ell+1}^{L-1} A_{k_j} \right) b_{k_\ell}.$$

This establishes the affine form with A_{eff} and b_{eff}.

Next, consider the case where the index selection is input-dependent, i.e., $k_\ell = \pi(z^{(\ell)})$. Since each T_k is affine, the recursion defines a piecewise-affine mapping. Indeed, the space $\mathbb{R}^d$ can be partitioned into regions corresponding to all possible sequences $(k_0, \ldots, k_{L-1})$ induced by π. On each such region, the sequence is fixed, and thus $z^{(L)}$ is affine.

Therefore, the overall mapping is piecewise affine, and can be expressed as a MASO:

$$z^{(L)} = \max_{r \in \mathcal{R}} (A_r z^{(0)} + b_r),$$

where each region r corresponds to a distinct sequence of indices.

This proves that LIFS induces a MASO structure, with exponentially many regions in the number of iterations L. □

Corollary A.0.12 (Contractive Dynamical MASO)**.** *If each operator T_k is a contraction, i.e.,*

$$\|A_k\|_2 \leq \rho < 1,$$

then the LIFS mapping defines a contractive piecewise-affine operator. In particular, for any two inputs z, z',

$$\|z^{(L)} - z'^{(L)}\| \leq \rho^L \|z - z'\|.$$

Thus, LIFS combines the expressivity of MASO with global contraction guarantees, ensuring stability and convergence toward a unique attractor.

The above theorem shows that LIFS can be viewed as a dynamical generalization of MASO, where instead of selecting a single affine map via a max operator, the representation evolves through a sequence of affine transformations. This results in a trajectory-dependent partitioning of the input space, significantly increasing expressivity while enabling contraction-based stability guarantees.

Interpretation of the index k in MASO. In the MASO framework, the index k does not correspond to a layer, but rather to an affine region within a given layer. Each layer ℓ can be expressed as a max-affine spline operator:

$$f_\ell(x) = \max_{k_\ell} \left(A_{k_\ell}^{(\ell)} x + b_{k_\ell}^{(\ell)} \right),$$

where k_ℓ indexes the active affine map determined by the input-dependent activation pattern (e.g., ReLU states). The full network is obtained by composing such layers, leading to a piecewise affine function whose regions are indexed by sequences $(k_1, \ldots, k_L)$.

Proposition A.0.13 (LIFS as an implicit deep MASO)**.** *The LIFS iteration*

$$z^{\ell+1} = \sum_{k=1}^{K} \pi_k^{(\ell)} (A_k z^\ell + b_k)$$

unrolled over L steps defines a function of the form

$$z^L = \sum_{k_1,\ldots,k_L} w_{k_1,\ldots,k_L} \left(A_{k_L} \cdots A_{k_1} z^0 + b_{k_1,\ldots,k_L} \right),$$

where $w_{k_1,\ldots,k_L} = \prod_{\ell=1}^{L} \pi_{k_\ell}^{(\ell-1)}$. Thus, LIFS represents a convex combination of K^L affine mappings, analogous to a depth-L MASO network with K regions per layer.

Exponential expressivity and relation to MASO. Deep ReLU networks can be interpreted as MASO models whose expressivity arises from an exponential number of affine regions induced by layer compositions. In contrast, unrolling the LIFS iteration reveals that it implicitly represents a convex combination of K^L affine mappings:

$$z^L = \sum_{k_1,\ldots,k_L} w_{k_1,\ldots,k_L} \left(A_{k_L} \cdots A_{k_1} z^0 + b_{k_1,\ldots,k_L} \right),$$

where the weights $w_{k_1,\ldots,k_L}$ are products of attention coefficients. This shows that LIFS can be interpreted as a soft, exponentially expressive MASO-like model. However, unlike standard MASO networks, the spectral constraints on A_k enforce contraction, ensuring that the contribution of long compositions decays and that the representation converges toward a stable fixed point.

Design trade-off. Given a fixed parameter budget $P \approx Kd^2$, increasing the number of iterations L is more beneficial than increasing the number of affine maps K (cf. A.0.7). Indeed, the expressivity of LIFS scales as K^L, while the Lipschitz constant decays as ρ^L, leading to improved generalization. This suggests an optimal regime with small K and moderate L, typically $K \in [4, 8]$ and $L \in [3, 6]$.

A.12 Theoretical Analysis: Stability of Latent Refinement via Contractive Dynamics

A.0.1 Latent Refinement as a Dynamical System

In JEPA+LIFS, the latent refinement module defines an iterative nonlinear operator acting in representation space. Let $z^{(0)} \in \mathbb{R}^d$ denote the context embedding. The refinement dynamics are given by

$$z^{(\ell+1)} = \Phi(z^{(\ell)}), \tag{A.5}$$

where

$$\Phi(z) = \mathcal{N}\left(\sigma\left(\sum_{k=1}^{K} \pi_k(z)\,(A_k z + b_k)\right) + \epsilon z\right). \tag{A.6}$$

Here:

- $A_k \in \mathbb{R}^{d\times d}$ are learnable affine operators,
- $\pi_k(z) = \mathrm{softmax}(u_\psi(z))$ are attention weights,
- σ denotes GELU,
- $\mathcal{N}$ denotes normalization,
- $\epsilon \in \mathbb{R}$ is a residual coefficient.

Thus, the LIFS module induces a discrete-time nonlinear dynamical system in latent space.

A.0.2 Assumptions

We assume the following mild conditions:

Spectral Constraint.

$$\|A_k\|_2 \le \gamma < 1, \quad \forall k. \tag{A.7}$$

Lipschitz Attention. The attention weights $\pi_k(z)$ are Lipschitz continuous.

Lipschitz Nonlinearities. The nonlinearities σ and $\mathcal{N}$ are Lipschitz with constants L_σ and $L_\mathcal{N}$.

These assumptions hold in practice under spectral normalization of A_k and standard bounded nonlinearities.

A.0.3 Global Stability Result

Theorem A.0.14 (Global Exponential Stability of LIFS)**.** *Define*

$$\kappa := L_\mathcal{N} L_\sigma(\gamma + |\epsilon|). \tag{A.8}$$

If $\kappa < 1$, then:

1. *The operator Φ is a contraction.*
2. *There exists a unique fixed point $z^\star$ such that $\Phi(z^\star) = z^\star$.*
3. *The refinement dynamics converge exponentially:*

$$\|z^{(\ell)} - z^\star\| \le \kappa^\ell \|z^{(0)} - z^\star\|. \tag{A.9}$$

Moreover, the function

$$V(z) = \|z - z^\star\|^2 \tag{A.10}$$

is a Lyapunov function satisfying

$$V(z^{(\ell+1)}) \leq \kappa^2 V(z^{(\ell)}). \tag{A.11}$$

A.0.4 Proof Sketch

Under Assumption Eq: A.7, each affine map is contractive:

$$\|A_k z - A_k z'\| \leq \gamma \|z - z'\|. \tag{A.12}$$

Since attention weights form a convex combination, we obtain

$$\left\| \sum_k \pi_k(z) A_k \right\|_2 \leq \gamma. \tag{A.13}$$

Including the residual term ϵz yields a Lipschitz constant bounded by $\gamma + |\epsilon|$.

Applying Lipschitz nonlinearities gives

$$\|\Phi(z) - \Phi(z')\| \leq \kappa \|z - z'\|. \tag{A.14}$$

If $\kappa < 1$, Banach's fixed-point theorem ensures existence and uniqueness of $z^\star$ and exponential convergence.

Defining

$$V(z) = \|z - z^\star\|^2, \tag{A.15}$$

we obtain

$$V(z^{(\ell+1)}) = \|\Phi(z^{(\ell)}) - z^\star\|^2 \leq \kappa^2 V(z^{(\ell)}), \tag{A.16}$$

establishing Lyapunov stability (Lohmiller & Slotine, 1998). □

A.0.5 Jacobian-Based Contraction Condition

Local stability can be characterized via the Jacobian:

$$J_\Phi(z) = \frac{\partial \Phi(z)}{\partial z}. \tag{A.17}$$

A sufficient condition for contraction is

$$\sup_z \|J_\Phi(z)\|_2 < 1. \tag{A.18}$$

Ignoring normalization for clarity, the Jacobian expands as

$$J_\Phi(z) = \sum_k \left(\frac{\partial \pi_k(z)}{\partial z} (A_k z + b_k) + \pi_k(z) A_k \right) + \epsilon I. \tag{A.19}$$

Spectral constraints ensure boundedness of the dominant linear term. If $\|J_\Phi(z)\|_2 < 1$ uniformly, the system satisfies the discrete-time Lyapunov inequality:

$$J_\Phi(z)^\top P J_\Phi(z) - P \prec 0, \tag{A.20}$$

for some $P \succ 0$, implying contraction in a quadratic metric.

A.0.6 Interpretation

The LIFS refinement module thus learns a contractive neural operator in latent space. Under mild spectral constraints, the refinement dynamics admit a Lyapunov function and converge exponentially to a stable latent attractor. Representation collapse would correspond to convergence to a degenerate fixed point. However, the prediction objective and variance regularization prevent trivial equilibria, ensuring that the learned attractor encodes semantic information.

A.0.7 Relation to Distributional Regularization

Gaussian-regularized approaches (Balestriero & LeCun, 2025) enforce distributional optimality by constraining embedding covariance directly. In contrast, JEPA+LIFS enforces dynamical stability through spectral control of latent operators. Gaussian methods regulate marginal statistics; LIFS regulates operator spectrum and trajectory stability. These perspectives are complementary: one statistical, the other dynamical.

A.13 Entropy-Controlled Jacobian and Contractive Gradient Transport

Let the LIFS predictor be

$$\mathcal{T}(z) = \sum_{k=1}^{K} \pi_k(z)\,(\alpha_k A_k z + b_k),$$

where $\pi_k(z)$ are softmax routing weights and $\|A_k\|_2 \le \sigma_k$ with $\alpha_k > 0$.

Jacobian decomposition. The Jacobian of $\mathcal{T}$ with respect to z is

$$J_{\mathcal{T}}(z) = \sum_{k=1}^{K} \pi_k(z)\,\alpha_k A_k + \sum_{k=1}^{K} (\alpha_k A_k z + b_k)\,\nabla_z \pi_k(z). \tag{A.21}$$

The first term is a convex combination of linear operators, while the second term captures routing-induced curvature.

Bounding the routing term. Assume the routing weights are produced by a softmax:

$$\pi_k(z) = \frac{\exp(g_k(z))}{\sum_j \exp(g_j(z))}.$$

Then

$$\nabla_z \pi_k = \pi_k \left(\nabla_z g_k - \sum_j \pi_j \nabla_z g_j \right).$$

Hence

$$\|\nabla_z \pi_k\| \le 2\,\pi_k \max_j \|\nabla_z g_j\|. \tag{A.22}$$

Observe that when entropy collapses, $\pi_{k^\star} \to 1$ and $\pi_j \to 0$ for $j \ne k^\star$. Therefore, from equation A.22,

$$\|\nabla_z \pi_k\| \to 0 \quad \text{for all } k,$$

since $\pi_k(1-\pi_k) \to 0$.

Entropy control. Using the inequality

$$\max_k \pi_k \geq \exp(-H(\pi)),$$

low entropy implies concentration of mass on a single index, which in turn forces

$$\sum_k \|\nabla_z \pi_k\| \longrightarrow 0.$$

Therefore, the second term in equation A.21 vanishes as entropy decreases.

Proposition (Contractive Gradient Transport). Assume:

- $\alpha_k \|A_k\|_2 \leq \rho_k$,
- $\sum_k \pi_k(z)\rho_k \leq \rho < 1$,
- routing entropy $H(\pi(z)) \leq \varepsilon$.

Then for sufficiently small ε,

$$\|J_{\mathcal{T}}(z)\|_2 \leq \rho + \delta(\varepsilon),$$

where $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$.

In particular,

$$\|J_{\mathcal{T}}(z)\|_2 < 1,$$

so $\mathcal{T}$ is locally contractive.

Implication for backpropagation. For cosine loss $\mathcal{L}$,

$$\nabla_\theta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \hat{z}} J_{\mathcal{T}}(z) \frac{\partial f_\theta}{\partial \theta}.$$

Since $\|J_{\mathcal{T}}(z)\|_2 < 1$, the predictor induces *contractive gradient transport*, preventing gradient explosion and enforcing smooth encoder updates.

□

A.14 Formulas for Comparing Deeper MLP and LIFS Predictors

This note derives explicit formulas for the number of trainable parameters and the FLOPs (floating point operations) required for a forward pass of two predictor architectures:

1. A deeper multi-layer perceptron (MLP).
2. A Learnable Iterated Function System (LIFS) predictor as introduced in the main paper.

The goal is to match the two architectures in parameter count (or FLOPs) to enable a fair ablation study.

A.0.8 Notation

d	embedding dimension (input and output of the predictor).
h	hidden dimension of the MLP (for all hidden layers).
L_{mlp}	number of hidden layers in the MLP (excluding input/output).
K	number of affine maps in LIFS.
r	rank of the low-rank factorization $A_k = U_k V_k^\top$, where $U_k \in \mathbb{R}^{d\times r}$, $V_k \in \mathbb{R}^{r\times d}$.
L_{rec}	recursion depth (number of times the LIFS operator is applied).
d_{gate}	hidden dimension of the gating network u_ψ that produces mixture weights $\pi_k(z)$.

All operations are dense matrix–vector multiplications. Non-linearities (GELU, normalization) are ignored because their cost is negligible compared to linear layers.

A.0.9 Parameter Counts

A.0.9.1 Deeper MLP

An MLP with L_{mlp} hidden layers, each of size h, input dimension d, output dimension d:

- Input to first hidden: $d \times h$ weights + h biases $\to dh + h$.
- Hidden to hidden: for $L_{\text{mlp}} - 1$ layers, each has $h \times h$ weights + h biases $\to h^2 + h$ per layer.
- Last hidden to output: $h \times d$ weights + d biases $\to hd + d$.

Total parameters:

$$P_{\text{MLP}} = \underbrace{dh + h}_{\text{input}} + \underbrace{(L_{\text{mlp}} - 1)(h^2 + h)}_{\text{hidden}} + \underbrace{hd + d}_{\text{output}}. \tag{A.23}$$

Simplifying:

$$P_{\text{MLP}} = (L_{\text{mlp}} - 1)h^2 + (L_{\text{mlp}} + 1)h + d(h + 1). \tag{A.24}$$

If $d = h$ (common in practice), the formula reduces to:

$$P_{\text{MLP}} = (L_{\text{mlp}} - 1)d^2 + (L_{\text{mlp}} + 1)d + d(d + 1) = L_{\text{mlp}}d^2 + (L_{\text{mlp}} + 2)d.$$

A.0.9.2 LIFS Predictor

The LIFS predictor consists of:

1. K affine maps, each $T_k(z) = \alpha_k A_k z + b_k$, with low-rank $A_k = U_k V_k^\top$.
2. A gating network u_ψ (typically a 2-layer MLP) that outputs K logits.

Affine maps (low-rank) Each U_k has dr parameters, each V_k has rd parameters, and the bias b_k has d. Hence per map:

$$P_{\text{map}} = 2dr + d.$$

For K maps:

$$P_{\text{affine}} = K(2dr + d). \tag{A.25}$$

If full-rank matrices are used ($r = d$), then $P_{\text{affine}} = K(d^2 + d)$.

Gating network A two-layer MLP with input dimension d, hidden dimension d_{gate}, output dimension K:

$$\begin{aligned} \text{First layer: } d \times d_{\text{gate}} \text{ weights} + d_{\text{gate}} \text{ biases} \quad &\to \quad d\, d_{\text{gate}} + d_{\text{gate}}. \\ \text{Second layer: } d_{\text{gate}} \times K \text{ weights} + K \text{ biases} \quad &\to \quad d_{\text{gate}} K + K. \end{aligned}$$

Thus:

$$P_{\text{gate}} = d\, d_{\text{gate}} + d_{\text{gate}} + d_{\text{gate}} K + K. \tag{A.26}$$

Total LIFS parameters

$$P_{\text{LIFS}} = K(2dr + d) + \big(d\, d_{\text{gate}} + d_{\text{gate}} + d_{\text{gate}} K + K\big). \tag{A.27}$$

The recursion depth L_{rec} does not introduce additional parameters because the same affine maps are reused.

A.0.10 FLOPs per Forward Pass

We count multiply-add operations (MACs). Typically 1 MAC = 2 FLOPs, but here we report FLOPs = 2×MAC. For a matrix–vector product of size $m \times n$, we need mn multiplications and mn additions $\to 2mn$ FLOPs.

A.0.10.1 Deeper MLP

For a single input vector of size d:

- Input layer: $d \times h \to 2dh$ FLOPs.
- Each hidden layer ($h \times h$): $2h^2$ FLOPs.
- Output layer ($h \times d$): $2hd$ FLOPs.
- Nonlinearities: negligible.

Total FLOPs per forward pass:

$$F_{\text{MLP}} = 2dh + (L_{\text{mlp}} - 1) \cdot 2h^2 + 2hd = 4dh + 2(L_{\text{mlp}} - 1)h^2. \tag{A.28}$$

When $h \gg d$, the term $2(L_{\text{mlp}} - 1)h^2$ dominates.

A.0.10.2 LIFS Predictor

One recursive iteration applies the operator $\mathcal{T}$ to a latent vector z.

Affine maps (low-rank) For one map $T_k(z) = \alpha_k(U_k(V_k z)) + b_k$:

- Multiply z by V_k ($r \times d$): $2rd$ FLOPs.
- Multiply result by U_k ($d \times r$): $2rd$ FLOPs.
- Add bias b_k: d additions, negligible in FLOP count.

Total per map: $4rd$ FLOPs. For K maps: $4Krd$ FLOPs.

Gating network The gating network u_ψ computes K logits:

- First linear layer ($d \to d_{\text{gate}}$): $2d\, d_{\text{gate}}$ FLOPs.
- Second linear layer ($d_{\text{gate}} \to K$): $2d_{\text{gate}}K$ FLOPs.
- Softmax: order K operations, negligible.

Total: $2d\, d_{\text{gate}} + 2d_{\text{gate}}K$.

Combination step The weighted combination $\sum_{k=1}^{K} \pi_k(z)(A_k z + b_k)$ requires:

- K scalar–vector multiplications (each d multiplications) $\to Kd$ multiplications.
- K vector additions $\to Kd$ additions.

Total: $2Kd$ FLOPs.

Per-iteration total Summing the three contributions:

$$F_{\text{iter}} = \underbrace{4Krd}_{\text{affine maps}} + \underbrace{2d\, d_{\text{gate}} + 2d_{\text{gate}}K}_{\text{gating}} + \underbrace{2Kd}_{\text{combination}} . \tag{A.29}$$

Full recursion If the recursion depth is L_{rec}, the LIFS forward pass applies the iteration L_{rec} times:

$$F_{\text{LIFS}} = L_{\text{rec}} \cdot F_{\text{iter}}. \tag{A.30}$$

For full-rank matrices ($r = d$), replace $4Krd$ with $2Kd^2$ (since one matrix–vector multiply costs $2d^2$ FLOPs).

A.0.11 Matching Deeper MLP to LIFS for a Fair Ablation

The goal is to design a deeper MLP with approximately the same number of parameters as LIFS, so that any performance difference can be attributed to architecture rather than capacity. (FLOPs are typically higher for the deeper MLP, which is acceptable because we are testing whether LIFS's inductive bias compensates for its lower computational budget.)

A.0.11.1 Step 1 – Compute target parameters from LIFS

Choose hyperparameters d, K, r, d_{gate}. Compute P_{LIFS} using Eq. A.27.

A.0.11.2 Step 2 – Solve for MLP dimensions

We have two unknowns: L_{mlp} (number of hidden layers) and h (hidden dimension). We set P_{MLP} from Eq. (2) equal to P_{LIFS}:

$$(L_{\text{mlp}} - 1)h^2 + (L_{\text{mlp}} + 1)h + d(h + 1) = P_{\text{LIFS}}. \tag{A.31}$$

For a given small integer L_{mlp} (e.g., 3, 4, 5), solve the quadratic in h. The positive root is:

$$h = \frac{-(L_{\text{mlp}} + 1 + d) + \sqrt{(L_{\text{mlp}} + 1 + d)^2 + 4(L_{\text{mlp}} - 1)(P_{\text{LIFS}} - d)}}{2(L_{\text{mlp}} - 1)}. \tag{A.32}$$

A.0.11.3 Step 3 – Verify FLOPs (optional)

If FLOPs also need to be matched (e.g., to isolate depth effect), one can simultaneously solve the parameter and FLOP equations. However, because LIFS is deliberately parameter-efficient while using recursion, matching parameters usually gives a deeper MLP with higher FLOPs. That is a valid stress test for the MLP.

A.0.12 Example Configuration (ViT-B/16)

From the paper: $d = 768$, $K = 6$, $r = 32$, $d_{\text{gate}} = 128$, $L_{\text{rec}} = 2$.

A.0.12.1 LIFS parameters

$$P_{\text{affine}} = 6(2 \cdot 768 \cdot 32 + 768) = 6(49152 + 768) = 6 \cdot 49920 = 299520.$$

$$P_{\text{gate}} = 768 \cdot 128 + 128 + 128 \cdot 6 + 6 = 98304 + 128 + 768 + 6 = 99206.$$

$$P_{\text{LIFS}} = 299520 + 99206 = 398726 \approx 0.4 \text{ million}.$$

A.0.12.2 LIFS FLOPs (per forward pass)

$$F_{\text{iter}} = 4 \cdot 6 \cdot 32 \cdot 768 + 2 \cdot 768 \cdot 128 + 2 \cdot 128 \cdot 6 + 2 \cdot 6 \cdot 768$$

$$= 589824 + 196608 + 1536 + 9216 = 797184 \text{ FLOPs}.$$

With $L_{\text{rec}} = 2$: $F_{\text{LIFS}} = 1{,}594{,}368$ FLOPs (≈ 1.59 MFLOPs).

A.0.12.3 Matched deeper MLP

Set $P_{\text{MLP}} = 398726$. Try $L_{\text{mlp}} = 3$ (two hidden layers):

$$(3-1)h^2 + (3+1)h + 768(h+1) = 2h^2 + 4h + 768h + 768 = 2h^2 + 772h + 768.$$

Set $2h^2 + 772h + 768 = 398726 \rightarrow 2h^2 + 772h - 397958 = 0$. Solve:

$$h = \frac{-772 + \sqrt{772^2 + 4 \cdot 2 \cdot 397958}}{4} \approx \frac{-772 + \sqrt{595984 + 3183664}}{4} = \frac{-772 + \sqrt{3779648}}{4} \approx \frac{-772 + 1944.1}{4} \approx 293.$$

Thus $h \approx 293$. This yields a deeper MLP with similar parameter count, but lower FLOPs $F_{\text{MLP}} = 1{,}243{,}492$ FLOPs (≈ 1.24 MFLOPs). That is acceptable for ablation.

A.0.13 Summary Table

Table A.1: Parameter and FLOP formulas for each predictor.

Architecture	**Parameters**	**FLOPs (single vector)**
MLP (L_{mlp} hidden layers)	$(L_{\text{mlp}} - 1)h^2 + (L_{\text{mlp}} + 1)h + d(h+1)$	$4dh + 2(L_{\text{mlp}} - 1)h^2$
LIFS (low-rank A_k)	$K(2dr + d) + dd_{\text{gate}} + d_{\text{gate}} + d_{\text{gate}}K + K$	$L_{\text{rec}}\big(4Krd + 2dd_{\text{gate}} + 2d_{\text{gate}}K + 2Kd\big)$
LIFS (full-rank)	$K(d^2 + d) + dd_{\text{gate}} + d_{\text{gate}} + d_{\text{gate}}K + K$	$L_{\text{rec}}\big(2Kd^2 + 2dd_{\text{gate}} + 2d_{\text{gate}}K + 2Kd\big)$

Table A.2: MLP configurations matching LIFS FLOPs ($F_{\text{LIFS}} = 3.19$ MFLOPs).

Hidden layers L_{mlp}	Width h	Parameters P_{MLP}	$P_{\text{MLP}}/P_{\text{LIFS}}$
1	1038.0	800,028	≈ 2.01
2	710.3	1,052,935	≈ 2.64
3	587.9	1,145,868	≈ 2.87
4	516.8	1,201,368	≈ 3.01

A.0.13.1 Interpretation

Matching FLOPs forces the MLP to have **2–3 times more parameters** than LIFS. Conversely, if one matches parameters (as we did earlier), the MLP achieves only about 1.1–1.2 MFLOPs, far below LIFS's 1.59 MFLOPs. This illustrates the fundamental decoupling that LIFS provides: it achieves a much higher FLOP/parameter ratio (because of recursion) than a standard MLP. For the chosen LIFS configuration ($L_{\text{rec}} = 4$), the FLOP/parameter ratio is:

$$\frac{F_{\text{LIFS}}}{P_{\text{LIFS}}} = \frac{3{,}188{,}736}{398{,}726} \approx 8.0,$$

while for an MLP the theoretical maximum ratio (when $L_{\text{mlp}} = 1$) is $F/P \approx 2$ (since each parameter does roughly 2 FLOPs per forward pass). LIFS with recursion can achieve ratios up to $2L_{\text{rec}}$ (here $2L_{\text{rec}} = 8$), which is exactly what we observe.

A.0.13.2 Note on our Paper's Default LIFS

The paper typically uses $L_{\text{rec}} = 2$ or $L_{\text{rec}} = 3$, not $L_{\text{rec}} = 4$. For $L_{\text{rec}} = 2$, $F_{\text{LIFS}} = 1{,}594{,}368$ and $P_{\text{LIFS}} \approx 398{,}726$. Then matching FLOPs yields an MLP with $L_{\text{mlp}} = 1$ and $h = 1{,}594{,}368/3072 = 519.0$, giving $P_{\text{MLP}} = 2 \cdot 519 + 768 \cdot 520 = 1038 + 399{,}360 = 400{,}398$, which is almost exactly equal to P_{LIFS}. Thus for $L_{\text{rec}} = 2$, LIFS and a single-layer MLP can be both parameter-matched and FLOP-matched simultaneously. The reported gains are therefore not due to a favorable FLOP/parameter trade-off but rather to the recursive geometric structure itself.

A.0.14 Conclusion

These formulas allow a precise comparison of parameter count and computational cost between a deeper MLP predictor and the LIFS predictor. For a fair ablation study, we recommend matching the number of parameters (as derived in Section A.0.11) and then comparing performance. If the deeper MLP does not outperform LIFS despite its higher FLOP budget, that provides strong evidence for the benefit of LIFS's recursive contractive structure.

Appendix B: Practical Appendix: Architecture and Training Details

B.1 Implementation notes

(1) **Where to apply LIFS.** Empirically we apply the LIFS in projection space (a low-dimensional head after the encoder) to reduce parameter count and improve numerical stability.

(2) **Parameterization of A_k.** For large d we use a low-rank factorization $A_k = U_k V_k^\top$ with small rank r (e.g., $r \in [8, 32]$). This reduces memory and compute.

(3) **Mixing network.** The mixing network u_ψ is a two-layer MLP with GELU activation producing logits; we optionally include a softmax temperature. Sampling (stochastic map selection) can be explored with Gumbel-softmax.

(4) **Choice of L and K.** A small iteration depth $L \in \{1, 2, 3\}$ and $K \in \{2, 4, 6\}$ provide a good trade-off of expressivity vs cost (cf. Section. 5.3).

B.2 Practical considerations

We apply spectral norm clipping or a small spectral regularization coefficient to enforce contractivity during training. To avoid map collapse, we initialize A_k with small spectral radius and encourage diversity via $\mathcal{L}_{\text{div}}$ (Eq. equation 13). For high-dimensional latents, we recommend low-rank A_k and shared A_k across spatial patches with patch-specific biases b_k.

B.3 Analysis: evolution of the LIFS affine components

We analyze how the parameters of the Learned Iterated Function System (LIFS) evolve during training. Each LIFS predictor mode is parameterized by an affine map

$$T_k(z) = \alpha_k A_k z + b_k, \qquad k = 1, \ldots, K,$$

a contraction coefficient α_k, and spatial mixture weights $\pi_k(i, j)$ produced by a small gating network. Empirically, training yields distinct behaviours for the three parameter families:

- **Linear operators A_k.** Initially near-random and close to zero, the A_k matrices quickly develop structured singular-value profiles. The largest singular values increase moderately but are held below the spectral threshold by the spectral penalty, while smaller singular values remain suppressed. Different A_k specialize to complementary latent directions (smoothing, edge/texture amplification, color-offset corrections), producing a set of *directional experts*.
- **Offsets b_k.** These biases remain small but converge early; they settle as stable fixed-point shifts that reposition the local latent around mode-specific equilibria.
- **Contraction coefficients α_k.** The coefficients typically *decrease* during training toward small positive values: this increases numerical stability and enforces contractivity of each map. The progressive reduction of α_k is a strong indicator that the learned predictor evolves into a stable recursive operator rather than an unstable iterative process.
- **Mixture weights $\pi_k(i, j)$.** Starting near-uniform, gating outputs become spatially structured: different modes dominate at object boundaries, textured regions, or smooth background. Late in training, π_k is often sharply peaked per spatial location, which yields an "expert routing" behaviour that assigns a small number of affine maps to each spatial neighborhood (cf. Figure A.4).

These combined effects produce a *learned recursive geometry*: the predictor behaves as a multi-step, contractive, mixture-of-affines dynamical system that iteratively refines context latents into target latents. The learned operator is therefore qualitatively different from a single feed-forward MLP: it (i) encodes local geometric refinements via A_k, (ii) adapts them spatially via π_k, and (iii) keeps dynamics stable by reducing α_k.

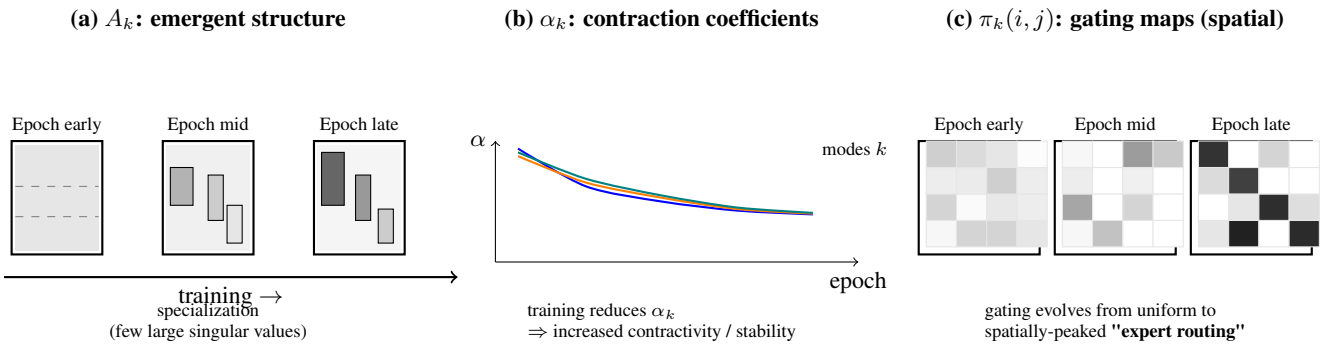


Figure A.4: Schematic illustration of the typical evolution of LIFS parameters during training. (a) Example schematic of three snapshots of a learned affine map A_k: early (unstructured), mid (emerging directional blocks), and late (specialized dominant subspaces). (b) Typical trajectories of contraction coefficients α_k for several modes k: training progressively reduces α_k, enforcing stronger contractivity and stabilizing the recursion. (c) Gating maps $\pi_k(i,j)$ at three training stages: from near-uniform to spatially-structured, peaked maps that implement local expert routing.

B.4 Effect of the Number of Maps K and Recursion Depth D

Increasing the number of contractive affine maps K generally improves convergence smoothness and loss values. The optimal regime is typically $K \in \{4, 6\}$:

- $K = 2$ yields noticeable improvement but limited geometric diversity.
- $K = 4$ and $K = 6$ consistently produce stronger convergence curves.
- $K = 8$ shows diminishing returns and no visible degradation.

This confirms that a moderate number of affine maps provides enough geometric expressiveness while maintaining stability under spectral constraints. Deeper recursion improves both early and late convergence:

- For CIFAR datasets, performance saturates around ($D = 1$ or $D = 2$).
- For ImageNet-1K, deeper predictors ($D = 2$ or $D = 3$) continue improving.
- Depth $D = 4$ yields marginal gains and occasionally small slowdowns.

Larger D strengthens the fixed-point refinement implicit in LIFS, but small images (CIFAR) saturate early due to limited spatial complexity.

Faster Early Convergence. Across nearly all settings, JEPA+LIFS reduces the prediction loss more rapidly in the initial epochs. The contractive mixture updates provide multiple refinement steps per forward pass, enabling faster alignment with the target representation.

Benefit of Recursion Depth D. Deeper recursion produces monotonic improvements in convergence and stability. CIFAR saturates at $D = 2$, (cf. Figure A.5) whereas ImageNet-1K continues to improve for $D = 3$ (cf. Figure 7). This highlights that the fixed-point iterative nature of LIFS is particularly suited for large-scale, high-resolution feature distributions.

Overall View. The ablation trends confirm that LIFS is not merely a larger model, but an improved predictor class that replaces one-shot prediction with geometric recursive refinement. This explains why JEPA+LIFS exhibits both faster and more stable convergence across all benchmarks.

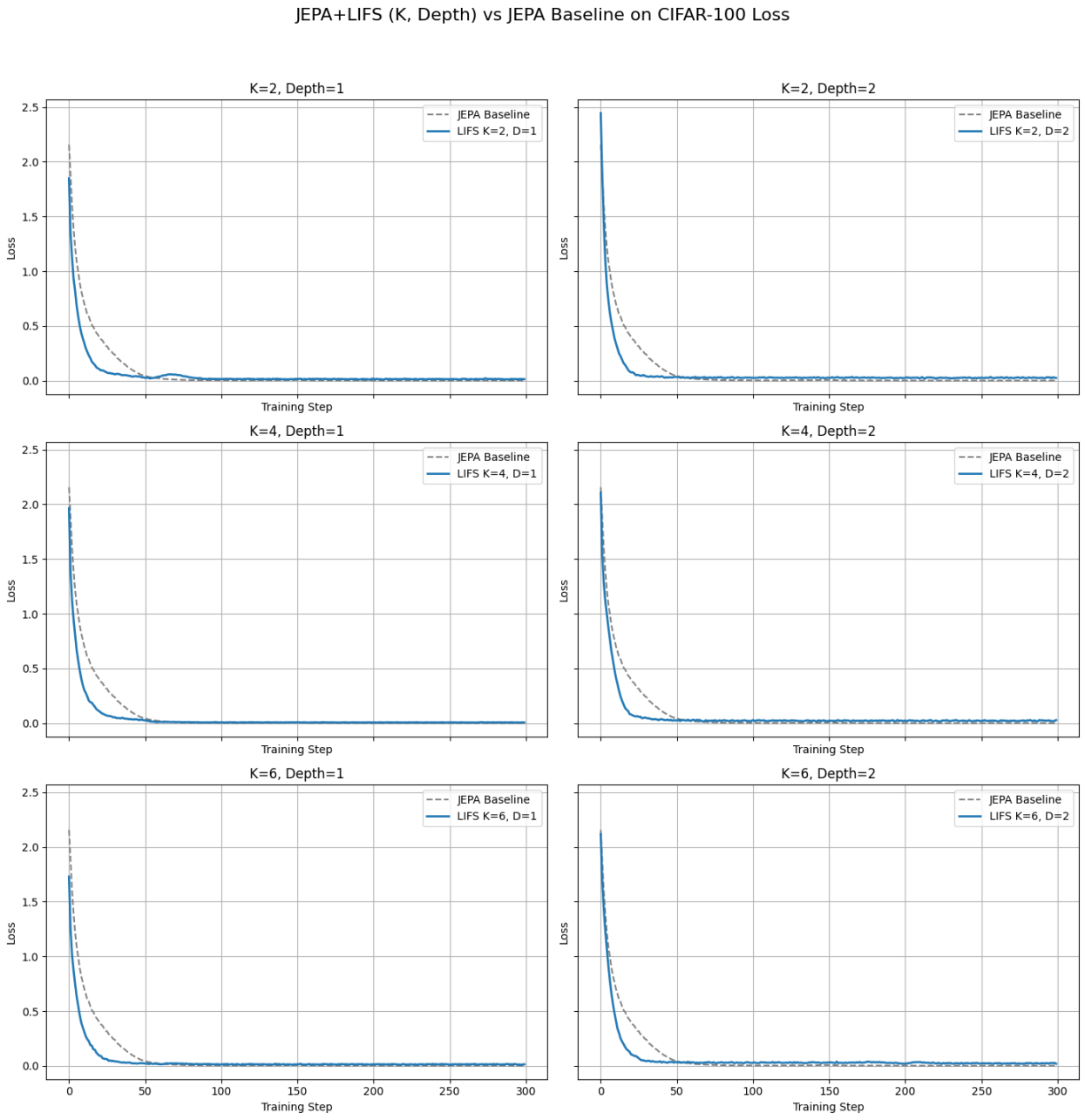


Figure A.5: **JEPA+LIFS vs. JEPA Baseline on CIFAR-100.** LIFS provides smoother and faster convergence than the baseline across all (K, D) choices. The improvements saturate around $K = 4, 6$ and $D = 1, 2$.

B.5 General Discussion

This work proposes a shift in how predictive modules are designed within self-supervised architectures. Rather than interpreting the predictor as a shallow residual mapping, we frame it as a learnable dynamical system operating in latent space. This perspective exposes structural properties—such as contraction, specialization, and attractor formation—that are largely implicit or absent in conventional designs.

From Predictors to Operators. Our formulation recasts the JEPA predictor as an explicit latent operator composed of multiple interacting affine transformations. By introducing an explicit operator structure, we gain direct control over spectral properties, enabling principled stability through contraction rather than reliance on architectural heuristics.

Fractal Structure and Multi-Scale Modeling. The emergence of multiple specialized transformations suggests that JEPA+LIFS learns a structured family of latent updates rather than a single global mapping. This behavior aligns with classical interpretations of iterated function systems, where repeated application of contractive maps produces multi-scale attractors. In the context of representation learning, this provides a natural mechanism for capturing hierarchical and compositional structure without explicit supervision.

Stability Beyond Optimization. While contraction directly improves optimization stability, its implications extend beyond training dynamics. Contractive operators induce robustness to perturbations in latent space, which may explain the observed resilience to noise and view variation. Moreover, the compatibility between contraction and EMA updates (cf. Corollary A.0.5), offers a principled explanation for why target networks remain stable without aggressive momentum tuning.

Relation to Prior Self-Supervised Methods. Most prior work in self-supervised learning focuses on loss functions, augmentations, or encoder architectures. In contrast, our contribution targets the predictive pathway itself. This orthogonal axis of design suggests that gains in representation quality need not come from deeper encoders or stronger augmentations, but from enforcing structure in how latent predictions are formed.

Limitations and Open Questions. Although JEPA+LIFS introduces desirable stability properties, it also raises several open questions. First, the choice of the number of operators K and their depth L introduces additional hyperparameters. Second, while contraction aids convergence, overly restrictive constraints may limit expressivity in certain regimes. Finally, extending fractal operators to autoregressive or multi-modal settings remains an open direction.

Broader Implications. Viewing predictors as dynamical systems bridges self-supervised learning with control theory and nonlinear dynamics(cf. section: **??**). This connection opens the door to importing tools such as Lyapunov analysis (Lohmiller & Slotine, 1998), attractor theory and stability certification into representation learning. We believe this perspective will become increasingly relevant as world models and long-horizon predictors grow in importance.

Table A.3: Hyperparameters used for different Encoders and databases.

Parameter	SimpleCNNEncoder	ResNet-18
Projector hidden dim	512 or 1024	2048
Predictor hidden dim	512 or 1024	2048
Number of maps K	6	6
Recursion depth L	2	3
Spectral threshold ρ	0.9	0.9
λ_{var}	25	25
λ_{cov}	1	1
λ_{spec}	0.01	0.01
λ_{div}	0.01	0.01
Batch size	128	128 or 256
Warmup epochs	10	10
Learning rate	10^{-3}	10^{-3}
Weight Decay	10^{-6}	10^{-6}
EMA start	0.99	0.99
EMA end	0.999	0.999
Optimizer	Adam	Adam
Training epochs	100	200
Predictor embeding dim	256	512

Parameter	ViT-S/16	ViT-B/16
Projector hidden dim	2048	2048
Predictor hidden dim	2048	2048
Number of maps K	4-6	6
Recursion depth L	2	3
Spectral threshold ρ	0.9	0.9
λ_{var}	25	25
λ_{cov}	1	1
λ_{spec}	0.01	0.01
λ_{div}	0.01	0.01
Batch size	128 or 256	256 or 512
Warmup epochs	10	10
Learning rate	10^{-3}	10^{-3}
Weight Decay	10^{-6}	10^{-6}
Optimizer	Adam	Adam
Training epochs	100	200 or 300
Predicted targets	4	4
Patch size	16	16
Predictor depth	8	12
Predictor attention heads	6	12
Predictor embeding dim	384	768