Reducing Data in Visual AI: I-JEPA

Optimizing I-JEPA for Data Efficiency

Maksim Plotnikov[1]

Supervisors: Jan van Gemert[1]**, Alex Manolache**[1]**, Petter Reijalt**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Maksim Plotnikov
Final project course: CSE3000 Research Project
Thesis committee: Jan van Gemert, Mitchell Olsthoorn

Abstract

Self-supervised learning eliminates the need for image labels to learn meaningful visual representations, but it does not remove the need for large pretraining datasets. This work studies how Image-based Joint-Embedding Predictive Architecture (I-JEPA) [1] behaves when pretraining data is deliberately limited. We train I-JEPA on stratified Tiny ImageNet [14] subsets and evaluate the frozen representations with CIFAR-10 [13] linear probing. The results show a steep improvement from the smallest subsets to the medium-data regime, followed by a plateau around the largest subsets under the standard final-checkpoint protocol. We also test two architectural modifications motivated by I-JEPA's design: reducing predictor capacity, to test whether an over-expressive predictor absorbs the pretext task instead of forcing useful encoder features, and adding shared photometric augmentation, to test whether extra input variation helps in low-data training. The shallow predictor improves transfer at 32k and 64k images but is neutral or harmful at the smallest and largest splits. The augmentation decreased downstream accuracy at 16k and was neutral at 32k. Additional controls—predictor depth sweeps, fixed-update budgets, and intermediate checkpoint analysis—suggest that the largest-split plateau is partly a training-dynamics issue rather than a pure data-efficiency ceiling. A cross-method comparison with Barlow Twins [16], MoCo [10], DINO [3], and MAE [9] under the shared protocol contextualizes I-JEPA's data efficiency among SSL alternatives.

1 Introduction

Self-supervised learning (SSL) has become a dominant way of training visual foundation models without the need for manual labels [11]. Instead of using annotated classes during pretraining, SSL methods construct their own targets from the image data. This is useful in computer vision where labels are particularly expensive, but it does not make the pretraining generally accessible. Nevertheless, many strong SSL results still depend on large proprietary or web-scale datasets [7]. For universities and smaller independent labs, the pretraining set itself can become the bottleneck.

This research studies that bottleneck for Image-based Joint-Embedding Predictive Architecture (I-JEPA) [1]. I-JEPA predicts masked image content in latent space: the model sees context regions of an image and learns to predict the representation of masked target regions. This makes it different from contrastive SSL methods like SimCLR [4] and MoCo [10], which often depend on chosen augmentations or negative samples, and from masked autoencoders like MAE [9], which reconstruct pixels. I-JEPA is therefore a reasonable candidate for low-data SSL. As of yet, its behavior at small pretraining scales still has to be measured.

While the original I-JEPA work focused on large-scale pretraining, more recent work has started to question whether SSL always needs large datasets [7]. Despite that, there is still limited evidence on how I-JEPA scales when the pretraining set is restricted. This work aims to address that gap using a controlled learning-curve experiment. An I-JEPA model is pretrained on restricted subsets of Tiny ImageNet [14] and each version is evaluated for downstream representation quality via linear probing on CIFAR-10 [13].

Specifically, this work aims to answer the research questions:

1. How does I-JEPA's downstream representation quality change as the amount of SSL pretraining data increases in the low-data regime?
2. How does I-JEPA compare to other SSL methods in the shared benchmark, including Barlow Twins [16], MoCo [10], DINO [3], and MAE [9]?
3. Which simple I-JEPA-specific changes, if any, improve transfer in this setting?

All three research questions are addressed in Section 4: the data-scaling curve (RQ1), the cross-method comparison under a shared evaluation protocol (RQ2), and the I-JEPA-specific architectural modifications (RQ3).

The remainder of this paper is organized as follows. Section 2 reviews related work and foundational SSL concepts. Section 3 describes the methodology, listing details about the pretraining, linear probing setup and evaluation metrics. Experiment design, motivation and results are presented in section 4, followed by a discussion of the findings in section 5. Section 6 addresses reproducibility and limitations, and Section 7 concludes with a summary and directions for future work.

2 Related Work

A Taxonomy of Visual Self-Supervised Learning

SSL in computer vision can be broadly categorized by the way models construct the pretext tasks without human labels. Understanding these paradigms reveals why data efficiency remains an open challenge.

- **Discriminative Methods:** This family trains models to maximize similarity between different augmented views of the same image. Contrastive approaches like SimCLR [4] and MoCo [10] rely on negative pairs to prevent representation collapse, while non-contrastive methods like BYOL [8], DINO [3], Barlow Twins [16], and VICReg [2] use teacher-student dynamics or regularization techniques to eliminate explicit negatives. While effective, these methods are notoriously data-hungry; their training signals depend heavily on diversity of heavily engineered data augmentations or large batch sizes, which can break down in low-data regimes [7].
- **Generative / Masked Image Modeling (MIM):** Popularized by Masked Autoencoders (MAE) [9], MIM hides patches of an image and tasks an encoder-decoder network with reconstructing the missing pixels. While such models provide a dense, localized training signal well-suited for Vision Transformers (ViTs) [6], pixel reconstruction forces the model to spend capacity on high-frequency, low-level details (such as background texture,

lighting variations) that are often irrelevant for downstream tasks [1].

I-JEPA and latent prediction

Joint-Embedding Predictive Architectures (JEPAs) predict missing information in representation space rather than observation space [5], making it stand out from the taxonomy above. I-JEPA applies this approach to images [1], and the same principle has been extended to other modalities such as audio [15]. The architecture consists of three core components: a context encoder, a target encoder (updated with Exponential Moving Average of the context encoder), and a predictor network. The latter takes the output of the context encoder alongside positional embeddings of the target masks, and maps them to the predicted target representations.

This does not imply that I-JEPA is automatically better suited to low-data learning, but makes it a peculiar candidate for research. I-JEPA's objective function removes the two pressure points present in other families: it does not require explicit negative samples, and it does not ask the model to reconstruct pixels.

Data efficiency in SSL

Many SSL papers report results after pretraining on ImageNet-scale or larger datasets. Those results are useful, but they make it hard to tell how much of the performance comes from the method and how much comes from scale. Recent small-data SSL work has started to address this issue directly [7]. This work follows the same motivation for I-JEPA: rather than evaluating one fixed data budget, we study the shape of the learning curve across several controlled pretraining budgets.

3 Method

This study adopts an empirical approach to reason about data efficiency of I-JEPA. We keep the evaluation protocol fixed, vary the amount of pretraining data, and then test a number of I-JEPA-specific changes to measure their impact on the learning curve.

Low-Data Pretraining Benchmark

To measure the data scale effect on representation quality, a controlled data-budgeting framework was built. For each pretraining budget n, a stratified subset D_n is sampled from Tiny ImageNet [14] and used to train an SSL model from scratch. The budgets are 1k, 2k, 4k, 8k, 16k, 32k, 64k, and 100k images. To isolate the effect from sampling variance, these datasets are nested ($D_{1\text{k}} \subset D_{2\text{k}} \subset ... \subset D_{100\text{k}}$) and class-stratified to ensure uniform class distribution across all budgets. We then freeze the encoder and evaluate the learned representations with a CIFAR-10 [13] linear probe. A method is deemed more data-efficient if it reaches higher downstream accuracy at the same budget, or if it reaches the same accuracy with fewer pretraining images.

To compensate for the low 64×64 resolution of Tiny ImageNet, our primary benchmark utilizes a ViT-Tiny/8 backbone [6] (depth 12, embed dim 192, 3 heads, MLP ratio 4). With patch size 8, each image becomes an 8×8 patch grid of 64 tokens. The baseline curves are established using ViT-Tiny/8, and the bottleneck effect is confirmed on ViT-Small/8 (depth 12, embed dim 384, 6 heads) in Section 4. Key hyperparameters are summarized in Section 8; the full configuration table is in Appendix 8.

I-JEPA Objective

I-JEPA divides an image into visible context regions and masked target regions. The context encoder processes the visible regions. The target encoder produces representations for the target regions. A predictor then uses the context representation and mask information to predict the target representation. The loss is computed between predicted and target representations, not between reconstructed and original pixels. The baseline masking configuration uses encoder mask scale $[0.5, 0.7]$, predictor mask scale $[0.1, 0.2]$, aspect ratio $[0.85, 1.2]$, one encoder mask, two predictor masks, no overlap, and a minimum of four kept tokens. With patch size 8 on 64×64 images, this operates over the 8×8 token grid. Full masking parameters are listed in Appendix 8.

The target encoder is updated as an exponential moving average of the context encoder. This gives the predictor a slowly changing target and reduces instability during training [8]. In the baseline protocol, each model is pretrained for 300 epochs. During pretraining, we log an inline Tiny ImageNet top-1 k-nearest-neighbor (k-NN) score as a proxy for representation quality, but the final metric is the downstream linear-probe result.

Downstream Evaluation

To assess representation quality without focusing too much on fine-tuning dynamics, linear probing approach was chosen. The pretrained encoder weights are frozen, and the final patch-token representations are mean-pooled to construct static vector representations of whole images in the dataset. These embeddings are then used to train a single linear classifier for 100 epochs. This evaluation is deliberately simple: it measures whether the SSL encoder already organizes features in a way that a linear classifier can use. Throughout this paper, we report the best validation accuracy achieved during the 100-epoch linear probe; this protocol is held fixed across all experiments. The only "best" versus "last" comparison we make refers to *pretraining* checkpoints: we compare the downstream performance of intermediate pretraining epochs against the final (epoch-300) pretraining checkpoint.

Inline Representation Diagnostics

To monitor how internal representations evolve over the course of pretraining, a suite of inline diagnostics is introduced, computed every 5 epochs:

- **Tiny ImageNet Top-1 k-NN Accuracy:** The unseen samples from Tiny ImageNet validation split are encoded with the latest encoder, and a simple k-NN classifier is trained on top. Its accuracy serves as a non-parametric proxy for downstream classification accuracy and is utilized for early-stopping experiments.
- **Effective Rank:** Measures the continuous dimensionality of the latent spaces. A sharp drop in effective rank

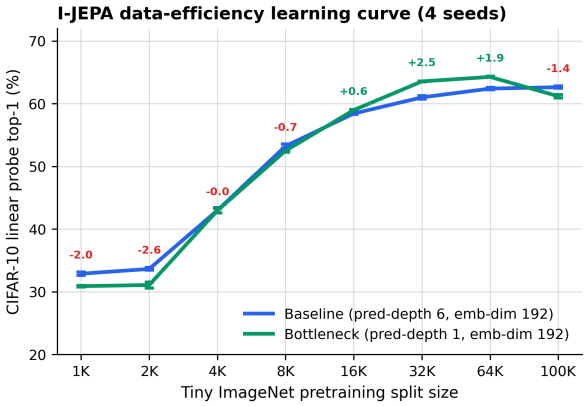

Figure 1: I-JEPA data-efficiency learning curve. CIFAR-10 linear probe top-1 accuracy (left) and inline Tiny ImageNet validation k-NN (right) as a function of pretraining data size, comparing baseline (pred-depth 6, emb-dim 192) and bottleneck (pred-depth 1, emb-dim 192) predictors. Delta labels show bottleneck minus baseline. Error bars are SEM across four seeds.

indicates dimensional collapse, where the model utilizes only a tiny subspace of its total capacity [12].

- **Participation Ratio:** Quantifies how evenly the variance of the representations is distributed across its principal components, acting as an indicator of feature diversity.
- **Top-1 Eigenvalue Share:** Tracks the fraction of total variance explained by the dominant eigenvector, allowing us to spot whether a single feature dimension is dominating the representation space.

4 Experiments and Results

This section reports the empirical evaluation of I-JEPA under constrained data budgets. First, a baseline data-scaling curve is established. This is followed by targeted experiments: a predictor depth sweep, an intermediate checkpoint analysis to understand training dynamics, a shared photometric augmentation test, and a fixed-update budget control. A cross-method comparison with Barlow Twins [16], MoCo [10], DINO [3], and MAE [9] contextualizes I-JEPA's data efficiency among SSL alternatives.

Baseline Learning Curve

Figure 1 shows the I-JEPA data-scaling result under the main ViT-Tiny/8 protocol. Accuracy improves sharply up to 16k images and continues more slowly until 64k. Notably, the 100k final checkpoint does not improve over 64k under the standard protocol, even slightly dropping. This plateau is closely mirrored by the inline Tiny ImageNet k-NN validation score.

Predictor Bottleneck

In the standard I-JEPA design, the predictor is the part of the model directly optimized to map context features to target features. A predictor with excessive capacity may "absorb" the pretext task: it minimizes the pretraining loss without forcing the encoder to learn transferable semantic features [1]. To test this, a sweep was executed across predictor depth and width.

Config	pred_depth	pred_emb_dim	Top-1
Baseline	6	192	61.54
Shallow+narrow	1	192	63.58
Wide baseline	6	384	60.17
Shallow+wide	1	384	63.49

Table 1: Factorial predictor ablation at 32k with ViT-Tiny/8. All values are CIFAR-10 linear probe top-1 (percentages).

Predictor depth	CIFAR-10 top-1
1	63.58
2	63.47
3	62.49
6	61.54

Table 2: Predictor depth sweep at 32k, ViT-Tiny/8, fixed width 192.

Predictor depth and width at 32k

At the 32k split, a factorial design tested four predictor configurations:

Table 1 shows two clear patterns. First, reducing predictor depth from 6 to 1 improves transfer regardless of width. Second, increasing width from 192 to 384 does not help; in fact, the wide baseline is worse than the narrow baseline. This supports the hypothesis that predictor depth, not width, is the relevant axis.

Predictor depth sweep at 32k

To test whether the effect is binary or gradual, a depth sweep was run at 32k with fixed width 192:

Table 2 shows a monotonic decline: depth 1 and 2 are essentially tied, while depth 3 is already about 1 percentage point worse, and depth 6 is the worst. This suggests that the useful axis is shallowness rather than an exact single-layer threshold.

Full learning curve with bottleneck

The shallow+narrow predictor (depth 1, width 192) was promoted to the full data-efficiency curve. Figure 1 shows that the bottlenecked predictor is not uniformly better. It improves clearly at 32k and 64k, is essentially neutral at 16k, and underperforms at the smallest and largest splits. The 100k underperformance is investigated next.

ViT-Small confirmation

To check whether the bottleneck effect transfers to a larger encoder, the same baseline and bottleneck configurations were run on ViT-Small/8 (depth 12, embed dim 384, 6 heads). Table 3 shows that the improvement pattern is consistent with ViT-Tiny/8 but with a larger effect: the bottleneck gains +6.23 pp at 32k and +3.05 pp at 64k, compared to +2.04 and +1.47 on ViT-Tiny/8. At 16k the bottleneck is positive (+2.68 pp) where ViT-Tiny/8 was neutral (-0.02 pp), suggesting the larger encoder benefits more from predictor capacity reduction at this split. At 100k the bottleneck is

Split	BL	BN	Δ
1k	32.29	29.32	−2.97
2k	32.44	30.45	−1.99
4k	42.83	44.04	+1.21
8k	54.46	52.38	−2.08
16k	58.84	61.52	+2.68
32k	62.58	68.81	+6.23
64k	67.01	70.06	+3.05
100k	69.29	70.28	+0.99

Table 3: ViT-Small/8 baseline (BL) vs bottleneck (BN) across splits. Values are CIFAR-10 linear probe top-1 (percentages). The bottleneck effect is larger than on ViT-Tiny/8 at 32k and 64k, and is positive at 16k where ViT-Tiny/8 was neutral.

Checkpoint	Top-1 (mean ± std)	Trend
ep50	62.63 ± 0.31	
ep100	$\mathbf{65.03 \pm 0.17}$	peak
ep150	64.29 ± 0.44	declining
ep200	63.16 ± 0.73	declining
ep250	61.73 ± 0.55	declining
ep300	61.17 ± 0.51	worst

Table 4: Intermediate checkpoint probes for the bottlenecked 100k run across four seeds. Values are CIFAR-10 linear probe top-1 mean and standard deviation (percentages). Epoch 300 is the final pretraining checkpoint; ep100 is the best. The peak at ep100 and monotonic decline thereafter is consistent across all seeds.

slightly positive (+0.99 pp) rather than negative, though the training-dynamics issue documented below for ViT-Tiny/8 was not investigated on ViT-Small. The shared photometric augmentation was also tested on ViT-Small at 16k, where it underperformed the baseline by 0.64 pp, confirming the negative augmentation result on a second backbone.

Intermediate Checkpoint Analysis

The weak 100k bottleneck result prompted an investigation of intermediate checkpoints. For the bottlenecked 100k run, checkpoints were saved every 50 epochs and probed individually. Table 4 reports the trajectory across four seeds.

Table 4 shows that the bottlenecked 100k run is actually strong, but only early: the best pretraining checkpoint is ep100 at 65.03%, which beats both the 100k baseline final checkpoint (62.51%) and the 64k bottleneck final checkpoint (64.10%). The last pretraining checkpoint (ep300, 61.17%) is the worst. The representation then degrades steadily. This peak-and-decline pattern is consistent across all four seeds, with ep100 having the tightest spread (std 0.17) and ep300 consistently lowest. The degradation is also visible in inline diagnostics: both CIFAR validation k-NN and Tiny ImageNet k-NN peak at the same checkpoint in the seed-0 instrumented run. Effective rank rises through ep150 and then declines, lagging the transfer peak—consistent with geometry metrics being explanatory rather than decisive (Section 4).

This finding shows that the standard final-checkpoint protocol can understate transfer for the largest split, and that training dynamics—not just data volume—matter for interpreting the bottleneck result at 100k.

Split	Baseline	Shared aug	Δ
16k	58.77	58.50	−0.27
32k	61.54	61.62	+0.08

Table 5: Shared photometric augmentation vs baseline, ViT-Tiny/8. Values are CIFAR-10 linear probe top-1 (percentages). Inline Tiny ImageNet k-NN also dropped at both splits.

Setting	Budget type	Top-1
Baseline 32k	fixed epochs	61.54
Baseline 32k	fixed updates	56.62
Bottleneck 32k	fixed epochs	63.58
Bottleneck 32k	fixed updates	57.92
Bottleneck 16k	fixed epochs	58.75
Bottleneck 16k	fixed updates	52.52

Table 6: Fixed-epoch vs fixed-update results. Fixed-update runs match the 100k/300e optimizer update count. Values are CIFAR-10 linear probe top-1 (percentages).

Photometric Data Augmentations

In non-predictive SSL frameworks like Barlow Twins [16], injecting input variations is essential to prevent representation collapse [12] and improve data efficiency without expanding the dataset. In this experiment, shared photometric transformations—color jitter (strength 0.25) and Gaussian blur—were applied globally before mask generation, so context and target blocks share the same altered color space. All other hyperparameters match the baseline ViT-Tiny/8 protocol.

Table 5 reports the results for 16k and 32k. At 16k, the augmented run underperforms the baseline by 0.27 pp on CIFAR-10 linear probe and also shows lower inline Tiny ImageNet k-NN. At 32k, the effect is essentially neutral (+0.08 pp). This suggests that standard photometric augmentations do not offer an easy data-efficiency recipe for I-JEPA. At 64×64 resolution, aggressive operations like Gaussian blur may destroy high-frequency spatial details that the latent prediction task relies on. It is also possible that I-JEPA, operating in representation space rather than pixel space, is simply less dependent on pixel-level distortions.

Same augmentations were additionally tested on ViT-Small at 16k, where it underperformed the baseline by 0.64 pp, confirming the negative augmentation result on a second backbone.

Fixed-Update Budget Control

The standard protocol uses 300 epochs for every split, which means smaller splits see far fewer optimizer updates. To decouple data volume from update count, fixed-update runs were created: 32k was trained for 944 epochs and 16k for 1888 epochs, approximately matching the 100k/300e update budget.

Table 6 shows that simply giving lower-data splits the same number of optimizer updates as 100k is harmful at the final checkpoint. Both baseline and bottleneck settings degrade by roughly 4–6 percentage points. However, the trajectory

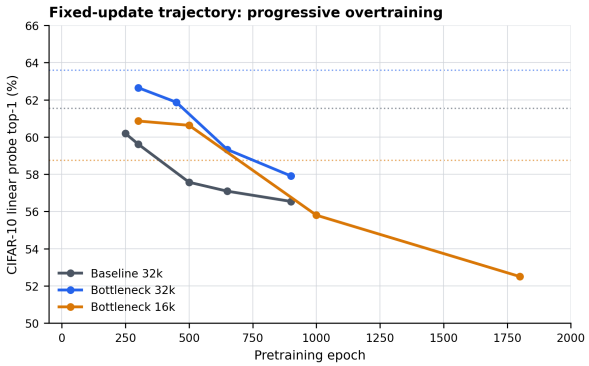

Figure 2: Fixed-update trajectory. CIFAR-10 linear probe top-1 across intermediate pretraining checkpoints for three fixed-update runs (Baseline 32k 944ep, Bottleneck 32k 944ep, Bottleneck 16k 1888ep). Dotted lines show the corresponding 300-epoch protocol result from Table 6. Checkpoints were selected at diagnostic-guided intervals, not regular spacing. All runs start near their 300-epoch anchor and decline progressively, indicating overtraining from extended training rather than a schedule mismatch.

in Figure 2 shows that this degradation is progressive, not sudden: all three runs start near their 300-epoch anchor (dotted lines) and decline steadily through extended training, confirming overtraining rather than a schedule mismatch.

Predictor Readouts and Geometry Diagnostics

To inspect why the shallower predictor leads to superior representations, two additional diagnostic quantities were measured during pretraining:

- **Ridge-Regression Predictor Gap:** A ridge regression is fit from the mean-pooled context-encoder representations to the mean-pooled target-encoder representations. The ridge gap is the difference between this ridge-proxy MSE and the JEPA predictor's own MSE on the same batch. A positive gap means the JEPA predictor achieves lower loss than the linear ridge baseline (indicating useful nonlinear capacity); a gap near zero means the predictor adds no nonlinear mapping beyond what a linear regression achieves.
- **Centered Kernel Alignment (CKA):** CKA measures the similarity of two representation spaces as a normalized Hilbert-Schmidt Independence Criterion. It ranges from 0 (no shared structure) to 1 (identical representational geometry). Here it is computed between the predictor's output and the target encoder's output, measuring how tightly the predicted latents align with the target latents in terms of their second-order structure.

Table 7 shows that the ridge gap between depth-1 and depth-6 predictors was negligible, suggesting this proxy is too coarse. CKA measurements showed that deeper predictors optimize for tighter latent alignment, while the shallow predictor consistently led to richer latent geometry (higher effective rank and participation ratio). The correlation between geometry and transfer is not perfect: effective rank can increase while downstream transfer worsens, so geometry metrics are treated as explanatory rather than decisive.

Metric	Depth 1	Depth 6
CIFAR-10 LP best (%)	**62.64**	62.13
Tiny-IN k-NN last (%)	15.51	**17.20**
CIFAR-val k-NN last (%)	49.02	**49.98**
Effective rank	**55.41**	47.12
Participation ratio	**36.82**	27.49
Pred–target CKA	0.757	**0.808**
Ridge gap	−0.00006	+0.00009

Table 7: Predictor diagnostics at 32k, comparing depth-1 and depth-6 predictors from 300-epoch instrumented runs. Bold marks the better value per row. The ridge gap (ridge-proxy MSE minus JEPA predictor MSE) is negligible for both depths, so it does not cleanly separate predictor capacity. CKA shows the deeper predictor achieves tighter predicted–target alignment without better downstream transfer.

Split	Config	Standard (300ep)	ES-best
16k	BL	58.77	58.77
16k	BN	58.75	60.37
32k	BL	61.54	60.45
32k	BN	63.58	62.18
64k	BL	62.63	60.10
64k	BN	64.10	62.66
100k	BL	62.51	61.46
100k	BN	60.56	61.92

Table 8: Early stopping results. "ES-best" is the CIFAR-10 linear probe top-1 of the checkpoint selected by inline Tiny ImageNet k-NN early stopping (patience 50). "Standard" is the 300-epoch final checkpoint from Figure 1. Values are percentages.

Early Stopping

The intermediate checkpoint analysis (Section 4) showed that the 100k bottleneck run peaks at epoch 100 and then degrades, raising the question of whether a validation-aware stopping criterion could automatically select a better checkpoint. To test this, an early stopping experiment was run on all splits for both baseline and bottleneck settings, training until the Tiny ImageNet validation k-NN score failed to improve for 50 consecutive epochs. The inline k-NN trajectory showed that larger splits (64k, 100k) stopped earlier than 300 epochs, while smaller splits (1k, 2k) often exhausted the patience window. Table 8 compares the downstream linear probe accuracy of the early-stopped "best" checkpoint (selected by inline k-NN) against the standard 300-epoch final checkpoint.

Table 8 shows that early stopping on the inline k-NN signal does not reliably improve downstream transfer. It hurts at most splits: the k-NN-selected checkpoint is worse than the 300-epoch endpoint for baseline at 32k, 64k, and 100k, and for bottleneck at 32k and 64k. The one notable exception is the 100k bottleneck (+1.36 pp), where early stopping partially recovers the training-dynamics degradation documented in Section 4. However, even this falls short of the ep100 intermediate checkpoint peak of 65.03%. The 16k bottleneck also improves by +1.62 pp, but this appears to be a small-data artifact rather than a general pattern. These results suggest that inline Tiny ImageNet k-NN is a weak proxy for

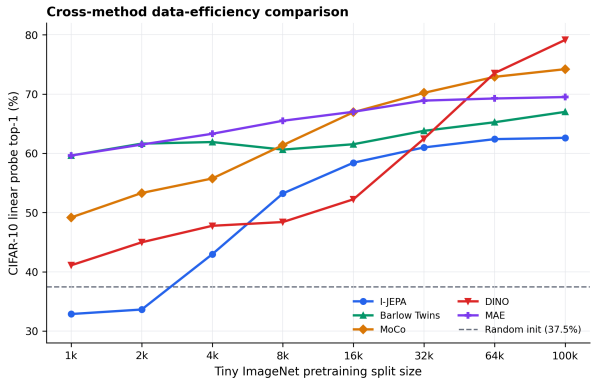

Figure 3: Cross-method data-efficiency comparison. CIFAR-10 linear probe top-1 as a function of Tiny ImageNet pretraining split size for six SSL configurations. I-JEPA curves are 4-seed means; Barlow Twins, DINO, and MAE are multi-seed means; MoCo is single-seed. The dashed line shows the linear probe accuracy of a randomly initialized ViT-Tiny/8 encoder, marking the floor below which performance is attributable to the probe rather than the encoder.

downstream CIFAR-10 transfer, consistent with the mismatch observed in Section 4 where inline k-NN favored the deeper predictor while transfer favored the shallower one.

Cross-Method Comparison

The comparison with Barlow Twins [16], MoCo [10], DINO [3], and MAE [9] under the shared protocol contextualizes I-JEPA's data efficiency. All methods are trained and evaluated under identical subsetting and probing rules; the results are presented in Figure 3.

Figure 3 shows that I-JEPA starts from near-random performance at the smallest splits (32.87% at 1k, below the 37.49% random-init floor) and improves steeply through the mid-range. The contrastive and masked-image methods—Barlow Twins, MoCo, DINO, and MAE—begin substantially higher at 1k and rise more gradually. All four comparison methods outperform I-JEPA across the full range, with DINO achieving the highest final accuracy (79.14% at 100k). The gap between I-JEPA and the other methods narrows at mid-to-large splits but does not close, suggesting that I-JEPA's latent-prediction objective has a slower data-efficiency ramp than contrastive or reconstructive approaches in this low-resolution, small-backbone regime.

5 Discussion

The I-JEPA data-scaling results support a nuanced conclusion. Under the standard final-checkpoint protocol, downstream accuracy improves sharply from the smallest subsets to 16k, continues more slowly to 64k, and then plateaus or slightly declines at 100k. This basic shape answers research question 1 in the affirmative: I-JEPA can learn transferable features from restricted pretraining data, but the standard recipe does not keep improving across the whole range tested.

Predictor bottleneck as a data-scale-dependent modification

Research question 3 asked whether simple I-JEPA-specific changes improve transfer. Figure 1 shows that the predictor bottleneck is the clearest positive signal, but it is not uniformly beneficial. The shallow+narrow predictor helps at 32k and 64k, is neutral at 16k, and hurts at the smallest and largest splits under the final-checkpoint protocol. This scale dependence is more informative than a uniform improvement would be: it suggests that the predictor capacity must be balanced against the encoder capacity and the data budget. At very small splits, the bottleneck may remove needed capacity; at intermediate splits, the baseline predictor may be too expressive relative to the encoder; at 100k, the latest checkpoint suffers from training dynamics that are addressed separately below.

The mechanism is most consistent with the "predictor absorbing the pretext task" hypothesis, but the evidence is indirect. Table 2 shows a monotonic relationship between depth and transfer at 32k, and Table 1 confirms that depth, not width, is the active variable. The ViT-Small/8 confirmation (Table 3) shows the effect is even larger on a bigger encoder, suggesting it is not specific to ViT-Tiny's capacity. However, Table 7 shows the ridge-regression capacity proxy did not cleanly separate depth-1 and depth-6 predictors, and CKA showed that deeper predictors achieve tighter alignment without better transfer. The paper therefore frames the bottleneck as a promising architectural control with supportive but not conclusive mechanistic evidence.

Training dynamics at the largest split

Table 4 shows that the 100k bottleneck result is an example of why training dynamics matter. Under the final-checkpoint protocol it is the worst point on the bottleneck curve, but intermediate checkpoint probing reveals a peak at epoch 100 (65.03%) that beats the 100k baseline and the 64k bottleneck. The representation then degrades steadily. This pattern is consistent across all four seeds. Inline diagnostics (CIFAR validation k-NN, Tiny ImageNet k-NN) peak around the same epoch in the seed-0 instrumented run, so the degradation is detectable during pretraining.

This effectively means that the standard 300-epoch protocol can understate transfer for the largest split, and that the plateau observed in the final-checkpoint curve is partly a checkpoint-selection issue rather than a pure data-efficiency ceiling. Whether a shorter schedule, a less aggressive late regularization, or a validation-aware early stopping rule can recover the 100k peak is still open.

Fixed-update controls and schedule sensitivity

Table 6 and Figure 2 show the fixed-update experiment designed to test whether the apparent data-efficiency curve is confounded by optimization budget. The result is negative in a useful way: simply giving smaller splits the same update count as 100k does not help; in fact, it hurts. The long runs peak mid-training and then degrade, suggesting that the current schedule (cosine LR decay, weight-decay ramp, EMA schedule) is not suited to repeated exposure of small datasets.

A more targeted follow-up—testing a smaller set of update multipliers rather than full 100k matching—would be needed to find any beneficial budget increase.

Augmentation

Table 5 shows that the shared photometric augmentation result is negative: the tested setting does not improve I-JEPA transfer at 16k and is neutral at 32k. This is consistent with the broader hypothesis that I-JEPA, operating in latent space rather than pixel space, has a different relationship to augmentation than contrastive methods do. It does not rule out milder augmentation variants, but the tested setting provides no positive signal to build on.

6 Responsible Research

Reproducibility

The experiments are run from explicit configuration files, and the Tiny ImageNet subsets are stored so that the same pretraining budgets can be reused across methods. Full hyperparameter settings, hardware details, dataset construction, random seeds, and checkpoint-selection rules are documented in Appendix 8. Code, configurations, and result data are publicly available.[1] The main learning curve (Figure 1) uses four seeds (0–3), each varying both the Tiny ImageNet data subsetting and the model initialization seed. The CIFAR-10 linear probe train/validation split is fixed at seed 0 across all runs. Single-seed runs (e.g., the predictor depth sweep, fixed-update control, and intermediate checkpoint analysis) are reported as point estimates, and trends that matter for the main claims are checked across multiple data budgets and related metrics to reduce seed-dependent noise.

Limitations

The study uses Tiny ImageNet [14] for pretraining and CIFAR-10 [13] for downstream evaluation. This gives a controlled benchmark, but it is still only one downstream task. It does not show whether the same representations transfer to detection, segmentation, medical images, or other domains.

The main protocol uses a small ViT-Tiny/8 backbone. That is appropriate for the compute budget and low-resolution images, but larger encoders may behave differently. The bottleneck improvement was confirmed on ViT-Small/8 (Table 3), where the effect is even larger at 32k and 64k, but the training-dynamics analysis at 100k was not repeated on the larger backbone.

The intermediate checkpoint trajectory for the bottlenecked 100k run (Table 4, Section 4) was probed across four seeds, confirming a consistent peak at ep100 and decline thereafter. However, the inline diagnostics (effective rank, k-NN) were only measured on the seed-0 instrumented run, so the geometry trajectory remains single-seed. More generally, small numerical differences should be treated cautiously unless they are repeated across seeds or supported by several related metrics. The strongest claims in the paper come from trends that are stable across data budgets, not from isolated differences of a few tenths of a percentage point.

[1] https://github.com/Mc-Seem/ijepa-data-efficiency

The cross-method comparison (Section 4) provides initial context, but the comparison methods were not exhaustively tuned, so the ranking should be read as indicative rather than definitive.

Early stopping on inline Tiny ImageNet k-NN was evaluated across all splits (Table 8), but the k-NN-selected checkpoints do not reliably improve downstream CIFAR-10 transfer, suggesting the inline proxy is a weak stopping signal for this task.

The project uses public image datasets and does not involve private or personally identifiable data. As with most work on visual representation learning, the same techniques could later be used in applications with social risks, including surveillance; this work does not deploy such systems, but the limitation is worth stating.

Use of Generative AI Tools

Generative AI tools were used during this project for coding assistance, writing feedback, and organizing the repository file structure. The specific tools used were Codex (OpenAI) for code generation and debugging, and Hermes (NousResearch) for writing feedback. All generated code and text was reviewed, tested, and edited by the author, who takes full responsibility for the final outcomes.

7 Conclusion and Future Work

This work studies I-JEPA as a low-data SSL method. The completed experiments address all three research questions: the data-scaling curve, the cross-method comparison, and the I-JEPA-specific modifications.

For research question 1, the ViT-Tiny/8 results show that I-JEPA learns transferable features from restricted Tiny ImageNet pretraining, with downstream CIFAR-10 linear probe accuracy rising steeply from 1k to 16k images, continuing more slowly to 64k, and then plateauing under the standard 300-epoch final-checkpoint protocol. The plateau is not necessarily a pure data-efficiency ceiling: the 100k bottleneck run peaks at epoch 100 and degrades thereafter, suggesting that training dynamics also limit the final-checkpoint result.

For research question 2, the cross-method comparison (Section 4, Figure 3) shows that I-JEPA has a slower data-efficiency ramp than Barlow Twins, MoCo, DINO, and MAE in this regime: all four comparison methods outperform I-JEPA across the full split range, starting substantially higher at 1k and maintaining a lead through 100k. DINO achieves the highest final accuracy at 79.14%, while I-JEPA plateaus around 62–64%.

For research question 3, the predictor bottleneck is the clearest positive modification, improving transfer at 32k and 64k by roughly 2 and 1.5 percentage points respectively on ViT-Tiny/8, with an even larger effect on ViT-Small/8 (Table 3). It is not uniformly beneficial: it is neutral at 16k and harmful at the smallest and largest splits under the final-checkpoint protocol. The depth sweep supports the interpretation that shallower predictors improve transfer, while the width ablation shows that predictor width is not the relevant axis. The shared photometric augmentation is a negative result at 16k and neutral at 32k. The fixed-update budget

control shows that simply matching optimizer updates across splits is harmful, pointing to schedule sensitivity rather than a raw update shortage. Early stopping on inline k-NN does not reliably improve downstream transfer, indicating that a better validation signal is needed.

Two conceptual directions are worth pursuing beyond the current scope. First, a data-aware pretraining schedule—one that adapts LR decay, weight decay, or EMA momentum to the dataset size—may recover the 100k peak without manual checkpoint selection. Second, the 8×8 patch grid imposed by Tiny ImageNet's 64×64 resolution may be a mismatch for the chosen I-JEPA masking parameters. Tuning the mask geometry for low-resolution pretraining is a concrete next experiment that would test whether the objective itself should be adapted to small-scale settings.

References

[1] 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. *arXiv*, 2023.

[2] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *International Conference on Learning Representations*, 2022.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[5] Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: A path towards autonomous machine intelligence. *J. Stat. Mech. (2024) 104011*, 2024.

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

[7] Carlos Vélez García, Miguel Cazorla, and Jorge Pomares. Escaping the big data paradigm in self-supervised representation learning. *arXiv*, 2025.

[8] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 2020.

[9] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked autoencoders are scalable vision learners. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *arXiv*, 2020.

[12] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations*, 2022.

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009.

[14] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. In *CS 231N Course Project Reports, Stanford University*, 2015.

[15] Ludovic Tuncay, Etienne Labbé, Emmanouil Benetos, and Thomas Pellegrini. Audio-jepa: Joint-embedding predictive architecture for audio representation learning. *ICME 2025, Jun 2025, Nantes, France*, 2025.

[16] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv*, 2021.

8 Hyperparameters and Reproducibility Details

Hardware and Software

All pretraining and evaluation runs were performed on a single workstation with an AMD Ryzen 7 5800X CPU and an AMD Radeon RX 9070 XT GPU (16 GB VRAM) under the ROCm software stack. Python 3.12.12 was used with PyTorch 2.9.1+rocm7.2.1, TorchVision 0.24.0+rocm7.2.1, NumPy 1.26.4, and Matplotlib 3.10.9. Dependencies were managed with uv via pyproject.toml.

Datasets

Tiny ImageNet [14] (200 classes, 64×64 resolution, $\sim$100k training images, $\sim$10k validation images) was used for pretraining. Stratified nested subsets of sizes 1k, 2k, 4k, 8k, 16k, 32k, 64k, and 100k were sampled. For the main learning curve, four data-subsetting seeds (0–3) were used; for single-seed experiments, seed 0 was used. Split indices are stored in the repository.[2]

[2] https://github.com/Mc-Seem/ijepa-data-efficiency

CIFAR-10 [13] (10 classes, 32×32 resolution) was used for downstream evaluation. The linear probe protocol uses a 10k-image training subset and a 5k-image validation subset, split with seed 0.

Model Architecture

The primary encoder is ViT-Tiny (depth=12, embed_dim=192, num_heads=3, mlp_ratio=4) with patch size 8, producing 64 tokens per 64×64 image. The baseline predictor has depth 6 and embed_dim 192. The bottleneck predictor has depth 1 and embed_dim 192. ViT-Small (depth=12, embed_dim=384, num_heads=6, mlp_ratio=4) was used for the confirmation runs in Section 4, with the same predictor configurations.

Pretraining Hyperparameters

Table 9 lists the baseline pretraining settings.

Hyperparameter	Value
Batch size	512
Base learning rate	1e-3
Start LR	2e-4
Final LR	1e-6
Warmup epochs	10
Weight decay	0.04
Final weight decay	0.4
LR schedule	Cosine decay with linear warmup
EMA momentum	[0.996, 1.0]
Epochs (standard)	300
ipe_scale	1.0
Mixed precision	bfloat16

Table 9: Baseline I-JEPA pretraining hyperparameters.

Masking. Encoder mask scale [0.5, 0.7], predictor mask scale [0.1, 0.2], aspect ratio [0.85, 1.2], num_encoder_masks=1, num_predictor_masks=2, allow_overlap=false, min_keep=4.

Augmentation. Random horizontal flip; random resized crop (scale [0.5, 1.0], size 64). No color distortion or Gaussian blur in the baseline.

Linear Probe Hyperparameters

Hyperparameter	Value
Frozen encoder	target_encoder
Head	Single linear layer
Batch size	256
Base LR	0.1 (scaled by batch size)
Epochs (protocol)	100
Optimizer	SGD with momentum 0.9
Weight decay	0.0
LR schedule	Cosine decay
Augmentation	None (cached features)
Best checkpoint	Selected by validation top-1

Table 10: CIFAR-10 linear probe hyperparameters.

Random Seeds and Known Caveats

The main learning curve uses four seeds (0–3), each varying both the Tiny ImageNet data subsetting seed and the model initialization seed. The CIFAR-10 linear probe train/validation split is fixed at seed 0 across all runs. Single-seed experiments (predictor depth sweep, fixed-update control, intermediate checkpoint analysis, shared augmentation, early stopping) are reported as point estimates. All main trends are checked across multiple data budgets and related metrics to reduce seed-dependent noise.

Configuration Files and Automation

Pretraining configurations and automation scripts are available in the repository.[3]

[3] https://github.com/Mc-Seem/ijepa-data-efficiency