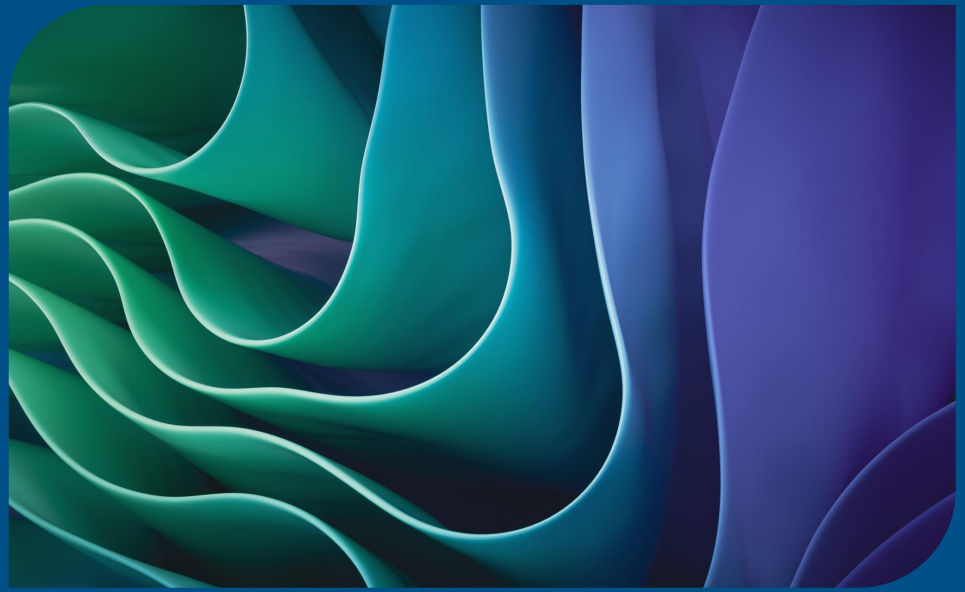
SimCLR & I-JEPA



1. Problem and SimCLR
2. Architecture
3. Contrastive Loss
4. I-JEPA
5. Architecture
6. Why I-JEPA?

Learning to See Without Labels: The SimCLR Approach

The Fundamental Limitation of Supervised Learning

The Label problem: Supervised learning, requires massive, manually labeled datasets.

For Example: ImageNet has 1.2 million human-annotated images.

The Cost: This process can be very expensive, slow, and simply doesn't scale to the vast amount of unlabeled data in the world.

Research Question the paper is on: How can a model learn meaningful visual representations directly from the data itself, without any human-provided labels?

The Human Analogy: Learning by Comparison

Imagine you are shown thousands of unlabeled images of cats and dogs.

We see that

A cropped, color-shifted, or blurred image of a cat is still fundamentally the same cat.

Any view of a dog is fundamentally different from any view of a cat.

our brains are performing a Contrastive Task:

It identifies two different transformations ("views") of the same image as a Positive Pair.

It recognizes that these views are more similar to each other than to views from any other image (Negative Examples).

SimCLR: Approach

The Fundamental Limitation of Supervised Learning

The network isn't trained to classify cats vs. dogs. Instead, it's given a Pretext Task: "Which of these many transformed patches came from the same original image?"

This task is the key: To solve it, the network must learn to ignore irrelevant noise (exact pixel position, absolute color) and **extract high-level, meaningful features** that are **invariant to those transformations.**

What it Learns: Concepts like shapes, edges, textures, and object parts—the fundamental building blocks of visual understanding.

Powerful Transferable Representations

After this **unsupervised pre-training** (no labels required!), the model has learned a powerful, **general-purpose feature extractor.**

For Downstream Tasks (e.g., Image Classification):

You can now take the pre-trained encoder -> freeze its weights, and -> simply train a single linear classifier on top of its features using a very small number of labeled examples.

This is known as Linear Evaluation and is the standard benchmark for representation quality.

Why it works? By solving a A. simple comparison task on unlabeled data, SimCLR B. learns representations so good that a linear classifier on top can rival the performance of a fully supervised model trained on millions of labels.

UP-Shot of whole Method

Problem: Supervised learning requires expensive labels.
Solution (SimCLR): Learn by comparing different views of your data.
Mechanism: A contrastive learning framework that pulls together similar images and pushes apart dissimilar ones.
Outcome: Creates a powerful, reusable visual feature extractor without needing any labeled data during pre-training.

1. Stochastic Data Augmentation Module

This module defines the predictive task.
For any input image **x**, we sample two augmentation operators **t ~ T** and **t' ~ T** from the same family of transformations **T** to create two correlated views:
1. $\tilde{x}_i$ **= t(x)**
2. $\tilde{x}$□ **= t'(x)**

This (i, j) is a **positive pair**.

- The composition of augmentations is **important**
  - The paper studies this and finds that **random cropping + strong color distortion** is the most crucial combination.

WHY?

- If you only use random cropping, many patches from the same image will have very similar color histograms (Figure 6).
- The network could *cheat* by simply matching colors instead of learning semantic features.
- Adding strong color distortion forces the network to look beyond color and learn more robust, generalizable features like **shapes and textures**.

Toy Example:

- **Image:** A picture of a red apple on a tree.
- **View 1 (t):** Randomly cropped to show mostly the apple, color jitter applied making it slightly less red.
- **View 2 (t'):** A different crop showing the apple and a leaf, converted to grayscale.

The network must learn that these two very different-looking patches are from the **same source image**.

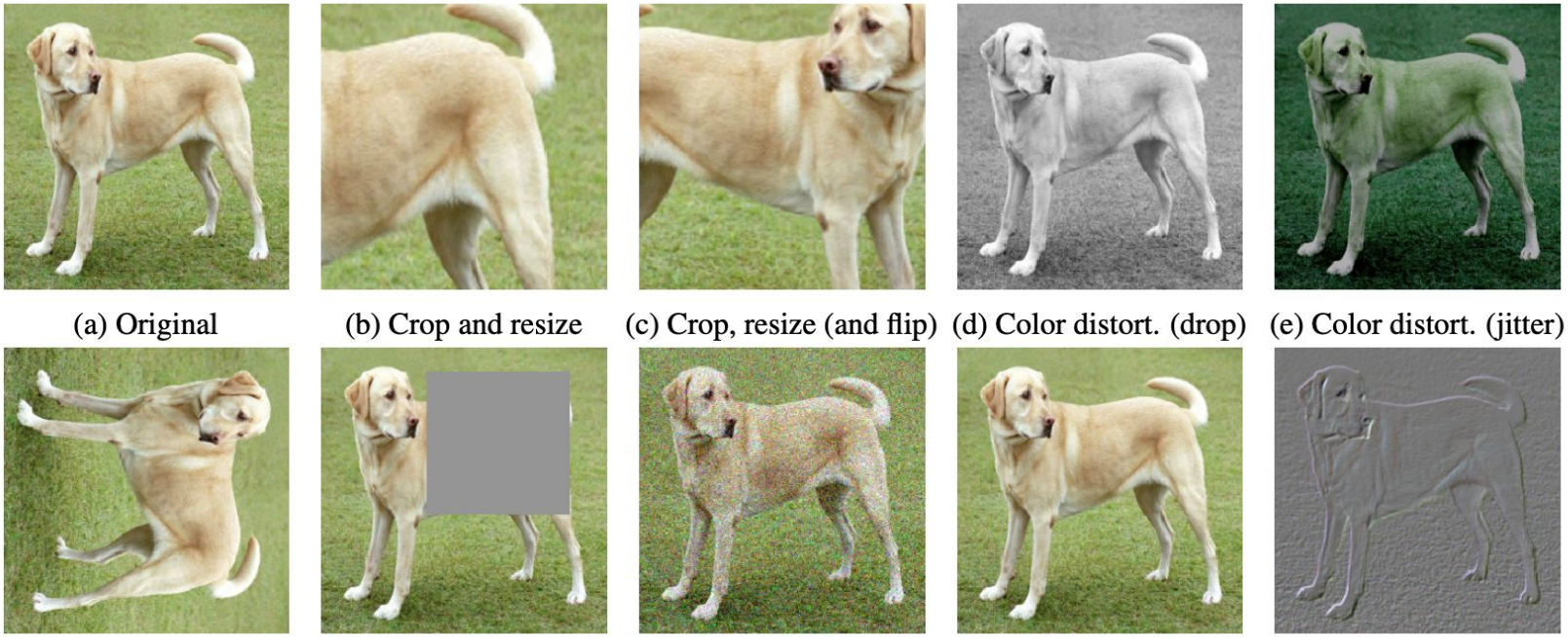
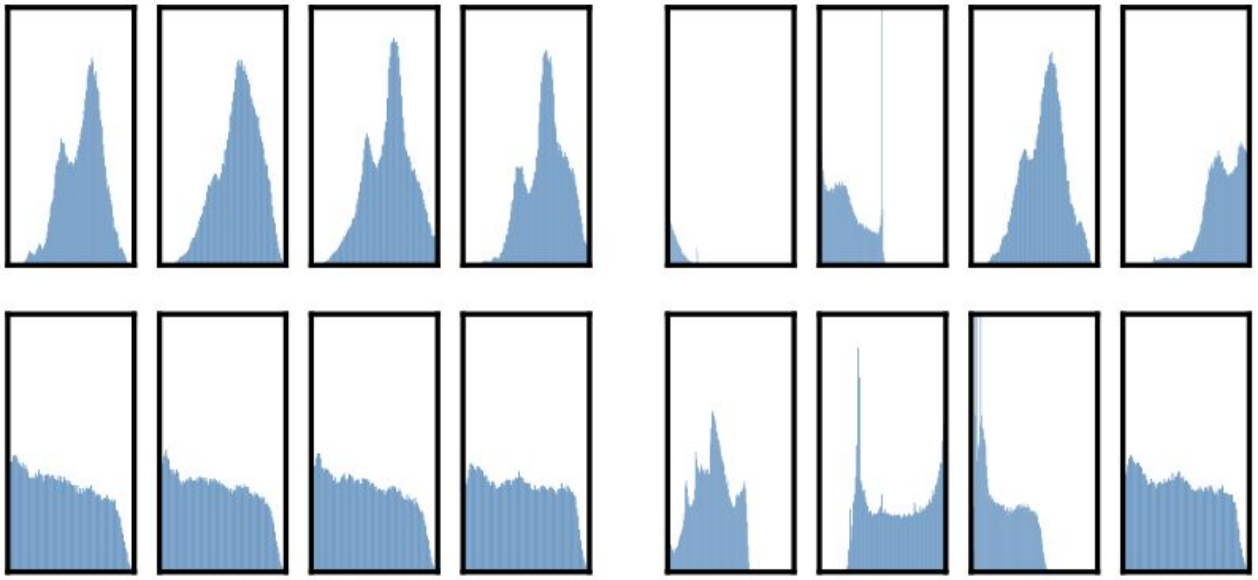


Figure 4. Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we *only* test these operators in ablation, the *augmentation policy used to train our models* only includes *random crop (with flip and resize)*, *color distortion*, and *Gaussian blur*. (Original image cc-by: Von.grzanka)



(a) Without color distortion. (b) With color distortion.

Figure 6. Histograms of pixel intensities (over all channels) for different crops of two different images (i.e. two rows). The image for the first row is from Figure 4. All axes have the same range.

2. Base encoder f(.)

1. Function and Purpose

- The base encoder **f(.) : R^(W×H×C) → R^d** is a parametric function (a neural network) that serves as the primary feature extractor.
- Its purpose is to learn a non-linear mapping from the high-dimensional pixel space of an input image $\tilde{\mathbf{x}}$ to a lower-dimensional, dense representation vector **h** That encodes the semantically relevant information of the input, invariant to the stochastic augmentations applied.
- It is trained to produce representations where the similarity structure in the output space **R^d** reflects the semantic structure of the input data, as dictated by the contrastive loss objective.

2. Architectural Specification

- **Input:** A tensor $\tilde{\mathbf{x}} \in \mathbf{R^{(224\times224\times3)}}$, which is a stochastically augmented view of an original image, resized to a standard spatial resolution.
- **Output:** A feature vector $\mathbf{h = f(\tilde{x}) \in R^d}$. For a standard ResNet-50 architecture, this is the output of the global average pooling layer, resulting in **d = 2048**.
- **Architecture Choice:** ResNet.

Role in the Contrastive Learning Framework

The encoder's parameters are learned by optimizing the contrastive loss defined on the outputs of the projection head $\mathbf{g(f(\tilde{x}))}$.

The gradient of the loss with respect to the encoder's parameters **θ_f** can be written conceptually as:

$$\text{Gradient of Loss} = (\partial L/\partial g) \times (\partial g/\partial f) \times (\partial f/\partial \theta\_f)$$

This means the encoder receives a learning signal that encourages it to produce representations **h** such that when projected (**z = g(h)**), the cosine similarity between positive pairs is maximized relative to negative pairs.

We conjecture that the importance of using the representation before the nonlinear projection is due to loss of information induced by the contrastive loss. In particular, $z = g(h)$ is trained to be invariant to data transformation. Thus, gcan remove information that may be useful for the downstream task, such as the color or orientation of objects. By leveraging the nonlinear transformation $g(\cdot)$, more information can be formed and maintained in h. To verify this hypothesis, we conduct experiments that use either h or g(h) to learn to predict the transformation applied during the pretraining. Here we set $g(h) = W^{(2)}\sigma(W^{(1)}h)$, with the same input and output dimensionality (i.e. 2048). Table 3 shows h contains much more information about the transformation applied, while g(h) loses information.

What to predict?	Random guess	Representation $\boldsymbol{h}$	Representation $g(\boldsymbol{h})$
Color vs grayscale	80	99.3	97.4
Rotation	25	67.6	25.6
Orig. vs corrupted	50	99.5	59.6
Orig. vs Sobel filtered	50	96.6	56.3

Table 3. Accuracy of training additional MLPs on different representations to predict the transformation applied. Other than crop and color augmentation, we additionally and independently add rotation (one of $\{0°, 90°, 180°, 270°\}$), Gaussian noise, and Sobel filtering transformation during the pretraining for the last three rows. Both $\boldsymbol{h}$ and $g(\boldsymbol{h})$ are of the same dimensionality, i.e. 2048.

4. Key Empirical Finding: Model Scaling

- A critical ablation study in the paper demonstrates that the quality of the learned representations, as measured by linear evaluation protocol, is highly dependent on the encoder's capacity.
  Observation:
- The performance gap between a linear classifier trained on self-supervised features and a fully-supervised model shrinks as the **encoder's capacity increases** (e.g., from ResNet-50 to ResNet-152, or by increasing width).

Interpretation:

- The contrastive learning objective is **a more demanding pre-training task** than supervised learning.
- It requires the model to **learn a complete representation of the data geometry without the simplifying cue of labels.**
- **A larger model capacity provides the necessary representational power** to solve this more complex pretext task effectively.
- This suggests that the **marginal benefit of additional parameters is greater for self-supervised learning than for its supervised counterpart.**

Implication:

- The effectiveness of contrastive learning is not just a function of the framework but is also constrained by the capacity of the encoder **f(.)**.
- State-of-the-art results necessitate large-scale models.

5. Distinction from the Projection Head g(.)

It is crucial to differentiate the roles of **f(.)** and **g(.)**:

- **The Encoder f(.)** is the permanent feature extractor.
  - Its output **h** is used for all downstream tasks after training.
- **The Projection Head g(.)** is a temporary, parametric transformation used only during contrastive pre-training.
  1. It maps representations to a space where the contrastive loss is applied optimally.
  2. Its purpose is to prevent loss of information in **h**; by allowing **g(h)** to become invariant to nuisances (e.g., augmentation artifacts), **h** is free to retain more semantically complete information.
  3. **g(.)** is discarded after pre-training.

Contrastive Loss Function (NT-Xent)

1. Objective and Necessity

The NT-Xent (Normalized Temperature-scaled Cross Entropy) loss is the objective function that trains the entire system.

- Its necessity comes from the core Theory of contrastive learning: **learning representations by contrasting positive pairs against negative pairs.**

Goal:

- Learn an embedding function (encoder f(.) + projection head g(.)) such that:
- Cosine similarity between projections (z_i, z_j) of a positive pair is maximized.
- Similarity between z_i and all other projections in the batch (negatives) is minimized.

Why this form?

- The loss must handle a large, changing set of negatives (each batch), remain numerically stable, and provide strong gradients, especially for **hard negatives** (negatives similar to the anchor but different instances).

2. Mathematical Deconstruction

Loss for a single positive pair (i, j):

$$\ell(i, j) = -\log \left[\exp(\text{sim}(z_i, z_j) / \tau) / \Sigma_{k=1..2N,\, k \neq i} \exp(\text{sim}(z_i, z_k) / \tau) \right]$$

a) Cosine Similarity

$$\text{sim}(u, v) = (u \bullet v) / (|u||v|)$$

- Purpose: Measures angular separation, invariant to vector magnitude.
- Why? Prevents cheating by simply increasing vector norms. Forces learning of meaningful directions in latent space.

b) Temperature Parameter τ

- Purpose: Scales similarity scores, sharpens or softens the probability distribution.
- Small τ (<1): sharpens, focuses heavily on hardest negatives → stronger gradient.
- Large τ (>1): softens, treats negatives more equally.
- Why? Proper τ ensures gradients are informative (not exploding, not vanishing).

Counterexample: why we need τ

Say anchor uuu has similarities with positives/negatives:

- u · v+=0.8u \cdot v_+ = 0.8u · v+=0.8 (positive).
- Negatives: [0.9, 0.3, 0.2].
- With **τ=1.0\tau=1.0τ=1.0:**
  exp(0.8)=2.22, exp(0.9)=2.46. The 0.9 negative is *stronger than the positive*! The loss punishes heavily.
- With **τ=0.1\tau=0.1τ=0.1:**
  exp(0.8/0.1)=exp(8)=2980, exp(0.9/0.1)=exp(9)=8103. Now the 0.9 negative utterly dominates the denominator, gradient focuses strongly on separating positive from this hard negative.

If τ\tauτ were too big (say 10), then exp(0.08)=1.083, exp(0.09)=1.094 — all values nearly identical, softmax nearly uniform → gradient signal weak.

So τ\tauτ is a knob for **relative hardness weighting**.

c) Softmax and Cross-Entropy Interpretation

- Numerator exp(sim(z_i, z_j) / τ) = score for positive class j.
- Denominator Σ_k≠i exp(sim(z_i, z_k) / τ) = sum of scores for all other examples.

So, probability P(j | i) = exp(sim(z_i, z_j) / τ) / Σ_k≠i exp(sim(z_i, z_k) / τ).

The loss ℓ(i, j) is just the negative log-likelihood of this probability.

d) Final Loss Calculation

- The full loss is averaged over all positive pairs (i, j) and (j, i):

L = (1 / 2N) Σ_k=1..N [ℓ(2k-1, 2k) + ℓ(2k, 2k-1)]

- Symmetry ensures every point serves as an anchor once → stable training.

Reason for SimCLR's success:

Relies on very large batch sizes, avoiding the need for a memory bank of negatives used in earlier methods.

Gradient Property:

- Harder negatives get larger weight → pushed away more strongly.
- Easy negatives contribute less.

Importance of Batch Size:

- Larger batch size → more negatives → better representations

I-JEPA: Self-Supervised Learning by Predicting Representations

The Core Problem: The Limitations of Prevailing SSL Paradigms

1. Invariance-Based Methods (SimCLR, DINO, MoCo):

Mechanism: Learn by making views of the same image (via hand-crafted augmentations) have similar embeddings.
Strength: Learn highly semantic features. Excellent for image classification.
Weakness: Introduce a strong inductive bias. The model is forced to be invariant to the chosen augmentations.

Why is this bad? This bias is often misaligned with downstream tasks. Object counting requires variance to scale and color. Depth prediction requires geometric information destroyed by cropping.

2. Generative Methods (MAE, BEiT):

Mechanism: Learn by reconstructing masked image content (pixels or tokens).
Strength: Less task-specific bias. More generalizable.
Weakness: Often get stuck on low-level details. Spending capacity to reconstruct high-frequency texture limits the semantic quality of the representations. Typically require full fine-tuning to compete.

The Research Question: Can we learn semantic representations without hand-crafted view augmentations while also avoiding the low-level trap of pixel reconstruction?

The I-JEPA soln: Prediction in Representation Space

Core Idea: From a single context block, predict the representations of multiple target blocks in the same image.

Why this works:

1. Avoids Invariance Bias: No distorted "views" are created. The context and target are natural parts of the image.

2. Avoids Low-Level Details: Predicting in representation space allows the target encoder to discard irrelevant pixel-level information (exact texture, lighting) and provide a abstract, semantic target for the predictor to aim for.

3. Encourages Semantic Learning: Predicting a large, missing region requires understanding the scene's semantic and geometric layout. You can't guess the coach's reaction without understanding the game.

Figure 4. Examples of our context and target-masking strategy. Given an image, we randomly sample 4 target blocks with scale in the range $(0.15, 0.2)$ and aspect ratio in the range $(0.75, 1.5)$. Next, we randomly sample a context block with scale in the range $(0.85, 1.0)$ and remove any overlapping target blocks. Under this strategy, the target-blocks are relatively semantic, and the context-block is informative, yet sparse (efficient to process).

The Architecture

1. The Target Encoder (f_bar_theta): The "Teacher" Network

- **What it is:** A standard Vision Transformer (ViT) that processes the entire, un-augmented, original image.
- **Input:** Full image y, patched into a sequence.
- **Output:** A dense set of patch-level representation vectors s_y = {s_y1, s_y2, ..., s_yN}. These are the regression targets.

Key Architectural Detail: Exponential Moving Average (EMA) Weights

- The weights (theta_bar) of the target encoder are not updated by gradient descent.
- They are updated as:
  theta_bar = lambda * theta_bar + (1 - lambda) * theta WHY?
  where (theta) are the weights of the context encoder and (lambda) is a momentum coefficient very close to 1 (e.g., 0.996).

Why this is Critical (The "Stop-Gradient")

- **Prevents Representational Collapse:** Collapse happens when the predictor and context encoder minimize loss trivially (e.g., outputting constant vectors). If both encoders were updated directly, they could co-adapt and collapse.
- **Provides Stable Targets:** EMA ensures the target representations (s_y) change slowly and smoothly. The predictor chases a moving but stable target, creating a "bootstrapping" effect where the student (context encoder) learns from the teacher (target encoder). WHY?
- **Conceptual Role:** Acts as a "memory" of past states of the context encoder, providing a consistent learning signal.

2. The Context Encoder (f_theta): The "Student" Network

- **What it is:** A ViT architecturally identical to the target encoder.
- **Input:** Only the subset of image patches specified by the context mask B_x (efficiency gain).
- **Output:** Patch-level representation vectors s_x for the visible context patches.

Key Detail: Gradient-Based Updates

- Its weights (theta) are the only encoder weights updated via backpropagation of the prediction error.
- **Learning Objective:** Produce a context representation (s_x) so semantically rich and spatially aware that the predictor can infer missing parts of the image. It learns a compressed, predictive world model.

3. The Predictor (g_phi): The "Abstract Reasoning" Module

- **What it is:** A separate, smaller network. The paper uses a narrow ViT (fewer channels) with significant depth.
- **Inputs:**
  1. Output of the context encoder (s_x) → semantic content.
  2. A set of mask tokens combined with positional embeddings. One token per patch position in the target block. Positional embedding provides spatial location for prediction.
- **Output:** Predicted patch-level representation (s_hat_y(i)) for a specific target block i.

Why its Design is Important

- **Bottleneck:** Limited capacity (smaller than encoder) prevents overfitting and forces learning of general mappings.
- **Conditioning on Position:** Predictor must know spatial position. Without position info, the task is ambiguous. Predictor learns g_phi(s_x, pos) = s_hat_y, i.e., the spatial-semantic relationships in images.
- **Role in Non-Collapse:** Predictor transforms context into predictions. This harder task (beyond maximizing similarity) helps avoid collapse. WHAT?

The Multi-Block Masking Strategy - Defining the Predictive Task

Target Blocks

- **Number:** M blocks (default 4). Predicting multiple diff blocks per image provides a richer, more varied learning signal within a single forward pass.
- **Size:** Large scale (15–20% of image area). This is the "semantic" key. A block this size likely contains a coherent object part (for example: a head, a wheel). Predicting it requires high-level understanding, not just texture synthesis.
- **Shape:** Random aspect ratio between 0.75 and 1.5. Good for generalisation since ensures the model is not biased towards squares.

Context Block

- **Size:** Very large scale (85–100% of image area). It must be highly informative. A small context would not help the model predict the target block with high accuracy
  The Key Removal Step: After sampling, any region of the context block that overlaps with any of the M target blocks is removed. This is very imp to ensure the prediction is non-trivial

Why?

- If the context and target overlap, the predictor could "cheat" by simply copying features from the overlapping region (from s_x to s_hat_y).
  This short-circuit would prevent the learning of a true predictive model.
  By removing overlap, the predictor is forced to perform extrapolation, not copy–paste.

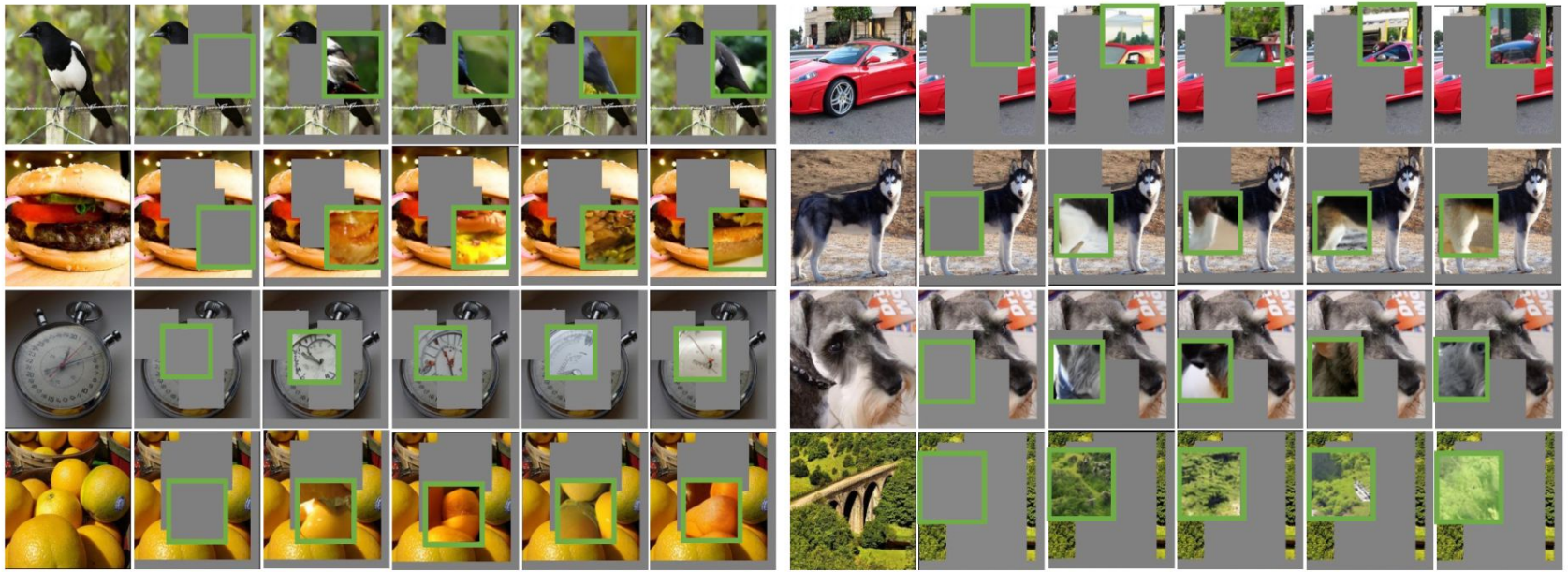


Figure 6. **Visualization of I-JEPA predictor representations**. For each image: first column contains the original image; second column contains the context image, which is processed by a pretrained I-JEPA ViT-H/14 encoder. Green bounding boxes in subsequent columns contain samples from a generative model decoding the output of the pretrained I-JEPA predictor, which is conditioned on positional mask tokens corresponding to the location of the green bounding box. Qualities that are common across samples represent information that contained is in the I-JEPA prediction. The I-JEPA predictor is correctly capturing positional uncertainty and producing high-level object parts with a correct pose (e.g., the back of the bird and top of a car). Qualities that vary across samples represent information that is not contained in the representation. In this case, the I-JEPA predictor discards the precise low-level details as well as background information.

Key Results & Why They Matter

1. Semantic Learning Without Augmentations (Table 1)

Linear Probing on ImageNet: I-JEPA (ViT-H/14) achieves 79.3%, significantly outperforming MAE (71.5%) and data2vec (77.3%) and approaching view-invariant methods like DINO (80.1%).

Takeaway: I-JEPA proves that hand-crafted augmentations are not necessary to learn highly semantic representations. Prediction is sufficient.

2. Superior Performance on Low-Level Tasks (Table 4)

Object Counting & Depth Prediction (Clevr): I-JEPA dramatically outperforms invariance-based methods (DINO, iBOT).

Why? I-JEPA is not forced to be invariant to spatial and color information. It retains the geometric and photometric cues essential for these tasks, which are actively discarded by methods like DINO.

3. Unmatched Computational Efficiency (Figure 5)

A ViT-H/14 trained with I-JEPA requires less compute than a ViT-S/16 trained with iBOT.
Why? I-JEPA only processes one view of the image (the context block). It does not need to generate and process multiple augmented views. Predicting in representation space is cheaper than decoding pixels.

4. Scalability

Performance improves with larger models (ViT-H → ViT-G) and larger datasets (ImageNet-1K → ImageNet-22K).

Takeaway: I-JEPA is not a niche method; it is a scalable, general approach.

Thank You