# A. Implementation Details

## A.1. Pretraining

**Architectures.** For I-JEPA pretraining, we use Vision Transformer [28] (ViT) architectures for the context-encoder, target-encoder, and the predictor. While the context-encoders and target-encoders correspond to standard ViT architectures, the predictor is designed as a light-weight (narrow) ViT architecture. Specifically, we fix the embedding dimension of the predictor to 384, while keeping the number of self-attention heads equal to that of the backbone context-encoder. For the smaller ViT-B/16 context-encoder, we set the depth of the predictor to 6. For ViT-L/16, ViT-H/16, and ViT-H/14 context-encoders, we set the depth of the predictor to 12. Finally, the ViT-G/16 uses a predictor of depth 16. I-JEPA is pretrained without a `[cls]` token. We use the target-encoder for evaluation and average pool its output to produce a global image representation.

**Optimization.** We use AdamW [50] to optimize the context-encoder and predictor weights. Our default batch-size is 2048, and the learning rate is linearly increased from $10^{-4}$ to $10^{-3}$ during the first 15 epochs of pretraining, and decayed to $10^{-6}$ following a cosine schedule thereafter. Following [3, 17], the weight-decay is linearly increased from $0.04$ to $0.4$ throughout pretraining. The target-encoder weights are identical to the context-encoder weights at initialization, and updated via an exponential moving average thereafter [3, 17, 22, 34, 36, 60]. We use a momentum value of $0.996$, and linearly increase this value to $1.0$ throughout pretraining, following [3, 17].

**Masking.** By default, we sample $4$ possibly overlapping target blocks masks with random scale in the range $(015, 0.2)$ and aspect ratio in the range $(0.75, 1.5)$. We sample $1$ context block mask with random scale in the range $(0.85, 1.0)$ and unit aspect ratio. We subsequently eliminate any regions in the context block mask that overlap with any of the $4$ target block masks. The context-block mask and target-block masks are sampled independently for each image in the mini-batch. To ensure efficient batch processing, we restrict the size of all context masks on a co-located GPU to be identical. Similarly, we restrict the size of all target masks on a co-located GPU to be identical. The mask-sampler is efficiently implemented in only a few lines of code in PyTorch [55] using a batch-collator function, which runs in the data loader processes. In short, in each iteration, the data loader returns a mini-batch of images and a set of context and target masks for each image, identifying the patch indices to keep for the context and target views.

## A.2. Downstream Tasks

**Linear evaluation.** When evaluating methods such as iBOT [75], DINO [17] or MAE [35], which leverage Vision Transformers [28] with an additional `[cls]` token, we use the default configurations of VISSL [33] to evaluate all the models on iNaturalist18 [63], CIFAR100 [44], Clevr/Count [41, 72], Clevr/Dist [41, 72], and Places205 [74]. We freeze the encoder and return the best number among the following representations: 1) the `[cls]` token representation of the last layer, 2) the concatenation of the last $4$ layers of the `[cls]` token. For each representation, we try two different heads: 1) a linear head, or 2) a linear head preceded by a batch normalization, and return the best number. We use the default data augmentations of VISSL [33]: random resize cropping and horizontal flipping, with the exception of Clevr/Count and Clevr/Dist, where we only use center crop and horizontal flipping, as random cropping interferes with the capability of counting objects and estimating distance, removing critical objects from the scene. For CIFAR100, we resize the images to $224 \times 224$ pixels, so as to keep the number of patches equal to that used during pretraining.

Because our I-JEPA implementation uses Vision Transformer architectures without a `[cls]` token, we adapt the default VISSL evaluation recipe to utilize the average-pooled patch representation instead of the `[cls]` token. We therefore report the best linear evaluation number among the following representations: 1) the average-pooled patch representation of the last layer, 2) the concatenation of the last $4$ layers of the average-pooled patch representations. We otherwise keep the linear-probing recipe identical.

**ImageNet evaluations.** To evaluate the I-JEPA on ImageNet [59], we adapt the VISSL recipe to use average pooled representations instead of the `[cls]` token. Following MAE [35], we use the LARS [70] optimizer with a batch-size of $16384$, and train the linear probe for 50 epochs. We use a learning rate with a step-wise decay, dividing it by a factor of $10$ every $15$ epochs, and sweep three different reference learning rates $[0.01, 0.05, 0.001]$, and two weight decay values $[0.0005, 0.0]$.

**Low-shot evaluation.** To evaluate our model on the ImageNet-1% low-shot task, we adapt the fine-tuning protocol of MAE [35].We fine-tune our ViT-L/H models for 50 epochs on ImageNet-1% with the AdamW optimizer and a cosine learning rate scheduler. We use a batch size of 512, a learning rate layer decay of 0.75 and 0.1 label smoothing. We use the default randaugment data-augmentations as in MAE. In contrast to the fine-tuning done with MAE, we do not use mixup, cutmix, random erasing or drop path. For the I-JEPA, we use a learning rate /weight decay of $3e^{-5}/5e^{-2}$ for the ViT-L/16, $3e^{-5}/4e^{-1}$ for the ViT-H/14 and $3e^{-5}/4e^{-1}$ for the ViT-H/$16_{448}$. Similar fine-tuning strategy for low-shot learning has been explored by Semi-VIT in the context of semi-supervised learning [15].

## B. Broader Related Work

Self-supervised learning of visual representations with joint-embedding architectures is an active line of research [2, 9, 11, 17, 22, 23, 34, 36, 53, 67, 75]. These approaches train a pair of encoders to output similar embeddings for two or more views of the same image. To avoid pathological solution, many popular joint-embedding approaches use explicit regularization [4,9,17,19] or architectural constraints [23,34]. Collapse-prevention based on architectural constraints leverage specific network design choices to avoid collapse, for example, by stopping the gradient flow in one of the joint-embedding branches [19], using a momentum encoder in one of the joint-embedding branches [34], or using an asymmetric prediction head [7, 19, 34]. Recent work [61] attempts to theoretically understand (in certain simplified settings) how joint-embedding methods with architectural constraints avoid representation collapse without explicit regularization.

Typical regularization-based approaches to collapse prevention in joint-embedding architectures try to maximize the volume of space occupied by the representations. This is often motivated through the InfoMax [51] principle. Indeed, a long-standing conviction in unsupervised representation learning is that the resulting representations should be both maximally informative about the inputs, while also satisfying certain simplicity constraints [32,49]. The former objective is often referred to as the information-maximization principle (InfoMax), while the latter is sometimes referred to as the parsimony principle [51]. Such approaches to representation learning have been proposed for decades (e.g., [13]), where, historically, simplicity constraints were enforced by encouraging the learned representations to be sparse, low-dimensional, or disentangled, i.e., the individual dimensions of the representation vector should be statistically independent [32]. Modern approaches enforce the simplicity constraints coupled with InfoMax regularization through self-supervised loss terms [5, 39, 40, 43, 54, 62]. One example is the widespread view-invariance penalty [52], often coupled with with independence [9, 71] or low-dimensionality constraints, e.g., by projecting representations on the unit hypersphere [19, 34, 36]. However, despite its proliferation, there have also been many criticisms of the InfoMax principle, especially since it is does not discriminate between different types of information (e.g, noise and semantics) [1]. Indeed, the sets of features we wish the model to capture are not always those with the highest marginal entropy (maximal information content).

Orthogonal to the contributions of invariance-based pretraining, another line of work attempts to learn representations by artificially masking parts of the input and training a network to reconstruct the hidden content [65]. Autoregressive models, and denoising autoencoders in particular, predict clean visual inputs from noisy views [7, 8, 18, 35, 65]. Typically, the goal is to predict missing inputs at a pixel level [28, 35, 68], or at a patch token-level, using a tokenizer [8, 66]. While these works demonstrate impressive scalability, they usually learn features at a low-level of semantic abstraction compared to joint-embedding approaches [3].

More recently, a set of approaches attempt to combine both joint-embedding architectures and reconstruction based approaches [29], wherein they combine an invariance pretraining loss with a patch-level reconstruction loss, as in the iBOT method [75]. Since view-invariance based approaches are typically biased towards learning global image representations, thereby limiting their applicability to other computer vision tasks, the idea is that adding local loss terms can improve performance on other popular tasks in computer vision [10, 25, 31]. The framework of contrastive predictive coding [54] is also closely related to this line of work on local loss terms. In the context of images [38], here the idea is to use a contrastive objective combined with a convolutional network to discriminate between overlapping image patch representations. Specifically, the goal is to encourage the representations of an image patch to be predictable of the image patches directly below it, while pushing away the representations of other patch views. In contrast to that work, the proposed I-JEPA method is non-contrastive and does not seek to discriminate between image patches. Rather, the goal is to predict the representations of various target blocks from a single context block. This is achieved with a Joint-Embedding Predictive Architecture, using a predictor network that is conditioned on positional embeddings corresponding to the location of the target block in the image. Qualitative experiments in Section 8 show that the predictor network in our architecture learns to correctly perform this local-to-local region feature mapping, and learns to correctly capture positional uncertainty in the image.

| Targets | | Context | |
|---|---|---|---|
| Scale | Freq. | Scale | **Top-1** |
| (0.075, 0.2) | 4 | (0.85, 1.0) | 19.2 |
| (0.1, 0.2) | 4 | (0.85, 1.0) | 39.2 |
| (0.125, 0.2) | 4 | (0.85, 1.0) | 42.4 |
| (0.15, 0.2) | 4 | (0.85, 1.0) | **54.2** |
| (0.2, 0.25) | 4 | (0.85, 1.0) | 38.9 |
| (0.2, 0.3) | 4 | (0.85, 1.0) | 33.6 |

Table 8. **Ablation of the target block size for multi-block masking**. Linear evaluation on 1% ImageNet-1K (using only 1% of the available labels); ablating the multi-block target size during I-JEPA pretraining of a ViT-B/16 for 300 epochs. Predicting larger (semantic) blocks improves the low-shot accuracy as long as the context is sufficiently informative.

| Targets | | Context | |
|---|---|---|---|
| Scale | Freq. | Scale | **Top-1** |
| (0.15, 0.2) | 4 | (0.40, 1.0) | 31.2 |
| (0.15, 0.2) | 4 | (0.65, 1.0) | 47.1 |
| (0.15, 0.2) | 4 | (0.75, 1.0) | 49.3 |
| (0.15, 0.2) | 4 | (0.85, 1.0) | **54.2** |

Table 9. **Ablation of the context size for multi-block masking**. Linear evaluation on 1% ImageNet-1K (using only 1% of the available labels); ablating the multi-block target size during I-JEPA pretraining of a ViT-B/16 for 300 epochs. Reducing the multi-block context size degrades the low-shot performance.

| Targets | | Context | |
|---|---|---|---|
| Scale | Freq. | Scale | **Top-1** |
| (0.15, 0.2) | 1 | (0.85, 1.0) | 9.0 |
| (0.15, 0.2) | 2 | (0.85, 1.0) | 22.0 |
| (0.15, 0.2) | 3 | (0.85, 1.0) | 48.5 |
| (0.15, 0.2) | 4 | (0.85, 1.0) | **54.2** |

Table 10. **Ablation of the targets number for multi-block masking**. Linear evaluation on 1% ImageNet-1K (using only 1% of the available labels); ablating the multi-block number of targets during I-JEPA pretraining of a ViT-B/16 for 300 epochs. Increasing the number of target blocks improve the low-shot accuracy.

## C. Additional Ablations

This section follows the same experimental protocol as Section 9. We report the result of a linear probe with a frozen backbone, trained on the low-shot 1% ImageNet-1K benchmark.

**Multiblock masking strategy.** We present an extended ablation of the multiblock masking strategy where we change the targets block scale (Table 8), the context scale (Table 9) and the number of target blocks (Table 10). We train a ViT-B/16 for 300 epochs using I-JEPA with various `multi-block` settings and compare performance on the 1% ImageNet-1K benchmark using a linear probe. In short, we find that it is important to predict several relatively large (semantic) target blocks, and to use a sufficiently informative (spatially distributed) context block.

**Masking at the output of the target-encoder.** An important important design choice in I-JEPA is that the target blocks are obtained by masking the *output* of the target-encoder, not the input. Table 11 shows the effect of this design choice on the semantic level of the learned representations when pretraining a ViT-H/16 using I-JEPA for 300 epochs. In the case where masking is applied to the input, we forward-propagate through the target-encoder once for each target region. Masking the output of the target-encoder during pretraining results in more semantic prediction targets and improves linear probing performance.

| Target Masking | Arch. | Epochs | Top-1 |
|---|---|---|---|
| Output | ViT-H/16 | 300 | **67.3** |
| Input | ViT-H/16 | 300 | 56.1 |

Table 11. **Ablating masking output of target encoder**. Linear evaluation on ImageNet-1K using only 1% of the available labels; ablating the effect of masking the target-encoder output during I-JEPA pretraining of a ViT-H/16 for 300 epochs. Masking the output of the target-encoder during pretraining significantly improves the linear probing performance of the pretrained representations.

**Predictor depth.** We examine the impact of the predictor depth on the downstream low-shot performance in Table 12. We pretrain a ViT-L/16 for 500 epochs using either a 6-layer predictor network or a 12-layer predictor network. The model pretrained using a deeper predictor shows a significant improvement in downstream low-shot performance compared to the model pretrained with a shallower predictor.

| Predictor Depth | Arch. | Epochs | Top-1 |
|---|---|---|---|
| 6 | ViT-L/16 | 500 | 64.0 |
| 12 | ViT-L/16 | 500 | **66.9** |

Table 12. **Ablating the predictor depth**. Linear evaluation on ImageNet-1K using only 1% of the available labels; ablating the effect of masking the predictor depth for a ViT-L/16 pretrained for 500 epochs. Increasing the predictor depth leads to significant improvement of the linear probe performance of the pretrained representations.

**Weight decay.** In Table 13, we evaluate the impact of weight-decay during pretraining. We explore two weight decay strategies: linearly increase the weight-decay from 0.04 to 0.4 or use a fix weight-decay of 0.05. Using a smaller weight decay during pretraining improves the downstream performance on ImageNet-1% when fine-tuning. However, this also leads to a degradation of performance in linear evaluation. In the main paper, we use the first weight decay strategy as it improves the performances in linear evaluation downstream tasks.

| Weight Decay | Arch. | Epochs | ImageNet-1% | ImageNet Linear-Eval |
|---|---|---|---|---|
| $0.04 \rightarrow 0.4$ | ViT-L/16 | 600 | 69.4 | **77.8** |
| 0.05 | ViT-L/16 | 600 | **70.7** | 76.4 |

Table 13. **Ablating the pretraining weight-decay**. We compare our default pretraining weight decay strategy where we linearly increase the weight-decay from 0.04 to 0.4 to using a fix weight decay of 0.05. Using a smaller weight-decay during pretraining can improve the fine-tuning performance on ImageNet-1%, However, it also leads to a drop of performance in linear evaluation.

**Predictor width.** We explore the impact of the predictor width in Table 14. We compare I-JEPA using a ViT-L encoder and a predictor with 386 channels to a similar model using a predictor with 1024 channels. Note that the ViT-L encoder has 1024 channels. Using a bottleneck in the predictor width improves the downstream performance on ImageNet 1%.

| Predictor Width | Arch. | Epochs | Top-1 |
|---|---|---|---|
| 384 | ViT-L/16 | 600 | **70.7** |
| 1024 | ViT-L/16 | 600 | 68.4 |

Table 14. **Ablating the predictor width.** We reports results on ImageNet-1K 1% using fine-tuning. We compare two predictors having a width of either 384 or 1024. Note the I-JEPA encoder is a ViT-L with 1024 channels. Having a width bottleneck in the predictor improves the downstream performances.

## D. Finetuning on the full ImageNet

In this section, we report performance on I-JEPA when fine-tuning on the full ImageNet dataset. We focus on the ViT-H/$16_{448}$ as this architecture achieves state-of-art performance with MAE [35].

We use a fine-tuning protocol similar to MAE. Specifically, we fine-tune our model for $50$ epochs using AdamW and a cosine learning rate schedule. The base learning rate is set to $10^{-4}$ and the batch size to $528$. We train using mixup set to $0.8$, cutmix set to $1.0$, a drop path probability of $0.25$ and a weight decay set to $0.04$. We also use a layer decay of $0.75$. Finally, we use the same rand-augment data-augmentations as MAE,

Table 15 reports the fine-tuning results. I-JEPA achieves $87.1$ top-1 accuracy. Its performance is less than $1\%$ away from the best MAE model despite I-JEPA being trained for $5.3$ times less epochs than MAE. This result demonstrates that I-JEPA is competitive when fine-tuning on the full ImageNet dataset.

| Method | Arch. | Epochs | Top-1 |
|---|---|---|---|
| MAE [35] | ViT-H/$14_{448}$ | 1600 | **87.8** |
| I-JEPA | ViT-H/$16_{448}$ | 300 | 87.1 |

Table 15. **Finetuning on the full ImageNet dataset**. I-JEPA achieves competitive performance. I-JEPA is close to MAE approach despite I-JEPA being trained for $5.3$ times less epochs than MAE.

## E. RCDM Visualizations

To visualize the representations of a pretrained neural network in pixel space, we use the RCDM framework [12]. The RCDM framework trains a decoder network $h_\omega$, comprising a generative diffusion model, to reconstruct an image $\boldsymbol{x}$ from the representation vector of that image $\boldsymbol{s}_x$ and a noisy version of that image $\hat{\boldsymbol{x}} := \boldsymbol{x} + \epsilon$, where $\epsilon$ is an additive noise vector. Concretely, the decoder objective is to minimize the loss function $\|h_\omega(\hat{\boldsymbol{x}}, \boldsymbol{s}_x) - \epsilon\|$. We train each RCDM network for 300,000 iterations using the default hyperparameters [12]. After training the decoder, one can subsequently feed the representation vector of an unseen test image $\boldsymbol{s}_y$ into the decoder along with various random noise vectors to generate several pixel-level visualizations of the representation, thus providing insight into the features captured in the representations of the pretrained network. Qualities that are common across samples represent information that is contained in the representation. On the other hand, qualities that vary across samples represent information that is not contained in the representations

In Figure 6, the visualizations are obtained by feeding the average-pooled output of the predictor, conditioned on a specific target region, into the decoder network, along with various random noise vectors. In Figures 7 and 8, the visualizations are obtained by feeding the average-pooled output of the target-encoder into the decoder network, along with various random noise vectors.

### E.1. Encoder Visualization

In Figure 7, we visualize the average-pooled I-JEPA representations at the output of our ViT-H/14 target-encoder. The first column contains the original image, while subsequent columns contain synthetic samples obtained by feeding the average-pooled representation of the image into the decoder along with various random noise vectors. Figure 7 suggests that the I-JEPA target-encoder is able to correctly capture the high-level information regarding objects and their poses, while discarding low-level image details and background information.

Figure 8 shows similar visualizations, but when using an MSN [3] pretrained ViT-L/7 target-encoder to compute the image representations. The MSN method trains a context- and target-encoder using a Joint-Embedding Architecture to enforce invariance of global image representations to various hand crafted data augmentations and missing patches. While the MSN pretrained network is able to capture high level semantic information about the image in the first column, it also exhibits higher variability in the generated samples, e.g., variability in the object pose, object scale, and number of instances. In short, the MSN pretrained discards much of the local structure in the image, which is in stark contrast to I-JEPA, which retains information about much of the local structure in the input image.

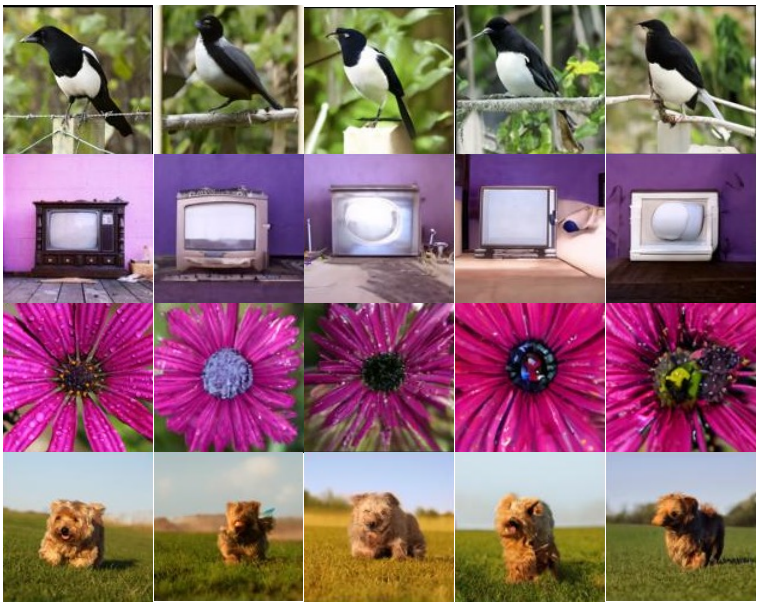

Figure 7. **Visualization of I-JEPA target-encoder representations**. For each image: first column contains the original image; subsequent columns contain samples from a generative model decoding the average-pooled output of a pretrained I-JEPA target-encoder. Qualities that are common across samples represent information that contained is in the I-JEPA representation. I-JEPA is able to correctly capture the high-level information regarding objects and their poses. Qualities that vary across samples represent information that is not contained in the representation. I-JEPA encoder discards the precise low-level details as well as background information.

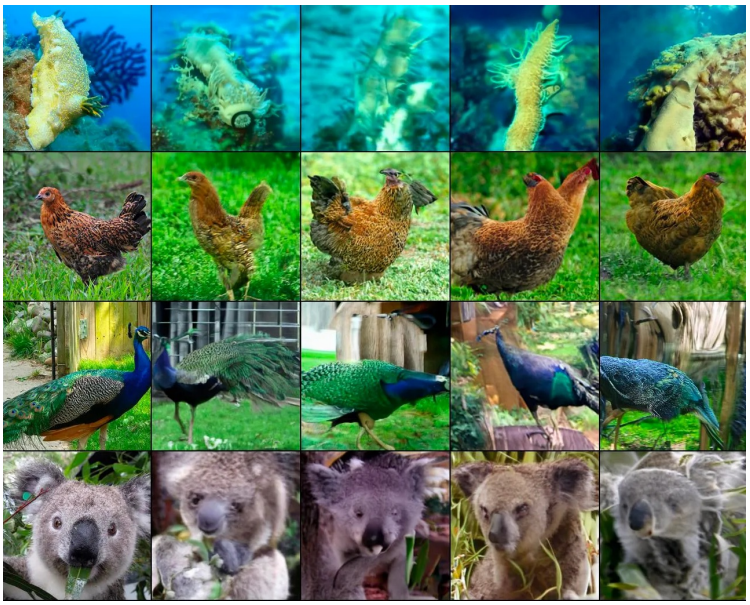

Figure 8. **Visualization of MSN target-encoder representations**. For each image: first column contains the original image; subsequent columns contain samples from a generative model decoding the output of a frozen MSN encoder [3]. Qualities that are common across samples represent information that is contained in the representation. Qualities that vary across samples represent information that is not captured by MSN. Compared to I-JEPA, MSN samples show higher variability. MSN retains less information from the input. In particular, it discards global structure information such as the object pose or even number of instances.