# Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture

**Authors:** Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, Nicolas Ballas

**Presenter:** Alec Xu

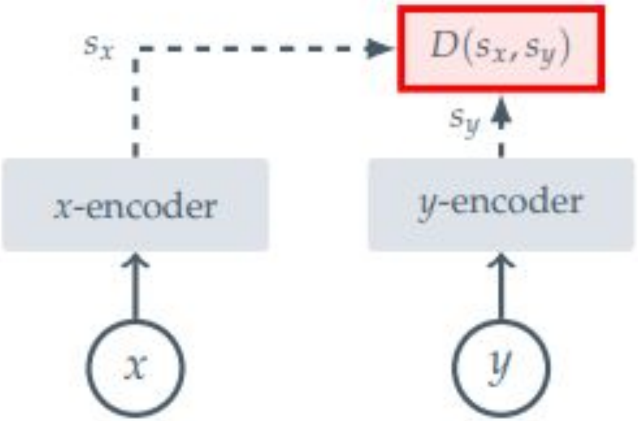
# Agenda

- Self-Supervised Learning
- Proposed Method
- Experiments and Results
- Ablation Studies
- Conclusion
- Quiz

# Self-Supervised Learning

- Type of representation learning
- Commonly used as a pre-training step before *downstream task*
  - Image classification, semantic segmentation, etc.
- "Self-supervised": labels generated from input themselves
  - No need to explicit annotations
- Types of SSL approaches
  - Joint-Embedding
  - Generative
  - Joint-Embedding Predictive

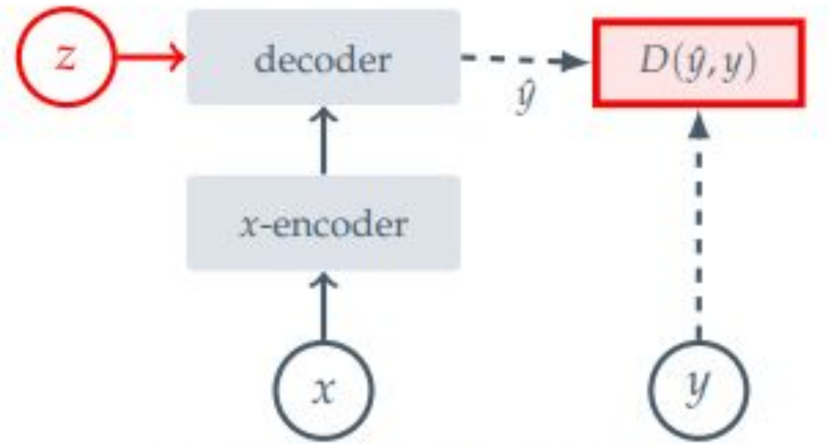
# Joint-Embedding Methods



(a) **Joint-Embedding Architecture**

- Inputs: images ***x, y***
- Goal: produce similar embeddings for similar ***x, y***, dissimilar embeddings for dissimilar ***x, y***
- Advantages
  - Highly semantic representations
- Disadvantages
  - Representation collapse
  - Hand-crafted image augmentations
  - Biases in downstream tasks
- **Example:** contrastive learning

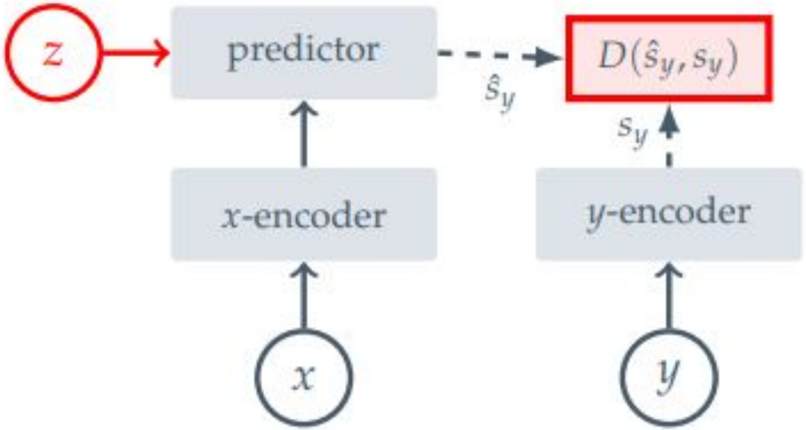
# Generative Methods




(b) **Generative Architecture**

- Inputs: images ***x*** and ***y***, additional (possibly latent) variables ***z***
- Goal: reconstruct ***y*** from ***x*** conditioned on ***z***
- Advantages
  - No representation collapse
  - Generalizable to other modalities
- Disadvantages
  - Worse semantic representations
- **Example:** reconstruct clean image ***y*** from masked version of ***x*** conditioned on (possibly learnable) mask ***z***

# Joint-Embedding Predictive Methods



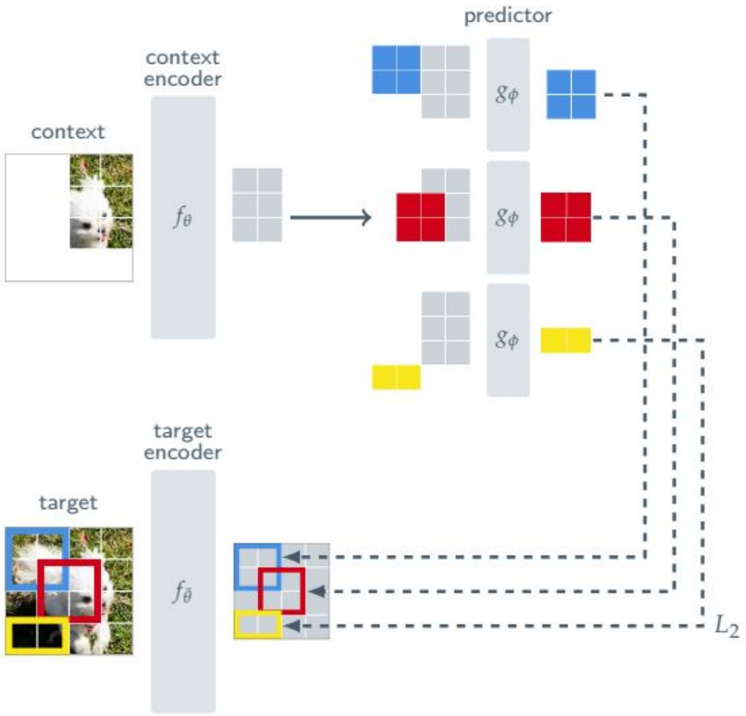
(c) **Joint-Embedding Predictive Architecture**

- Similar to generative approaches, but predictions are made in *embedding space*
- Advantages
  - Better semantic representations compared to generative methods
  - Still generalizable to other modalities
- Disadvantages
  - Representation collapse

# Proposed Approach: I-JEPA

- I-JEPA: **I**mage-based **J**oint-**E**mbedding **P**redictive **A**rchitecture
- Idea: given a *context block*, predict *target blocks*
  - Predictions in embedding space
- Networks: *context encoder*, *target encoder*, *predictor*
  - Encoders: Vision Transformers (ViT)
  - Predictors: Narrow ViT

# Proposed Approach: I-JEPA




1. Produce context block embeddings using *context encoder*
2. Predict *M* target block embeddings using a *predictor* conditioned on target block locations
3. Loss: average $L_2$ distance

- Updating weights
  - Context encoder and predictor: gradient-based optimization
  - Target encoder: exponential moving average of context encoder weights

# Proposed Approach: I-JEPA

- Better semantic representations compared to other approaches without augmentations
- No hand-crafted data augmentations
- Asymmetric architecture to avoid representation collapse

- How to choose target and context blocks?

# I-JEPA Target Blocks

1. Convert input image $\boldsymbol{y}$ into $N$ non-overlapping patches

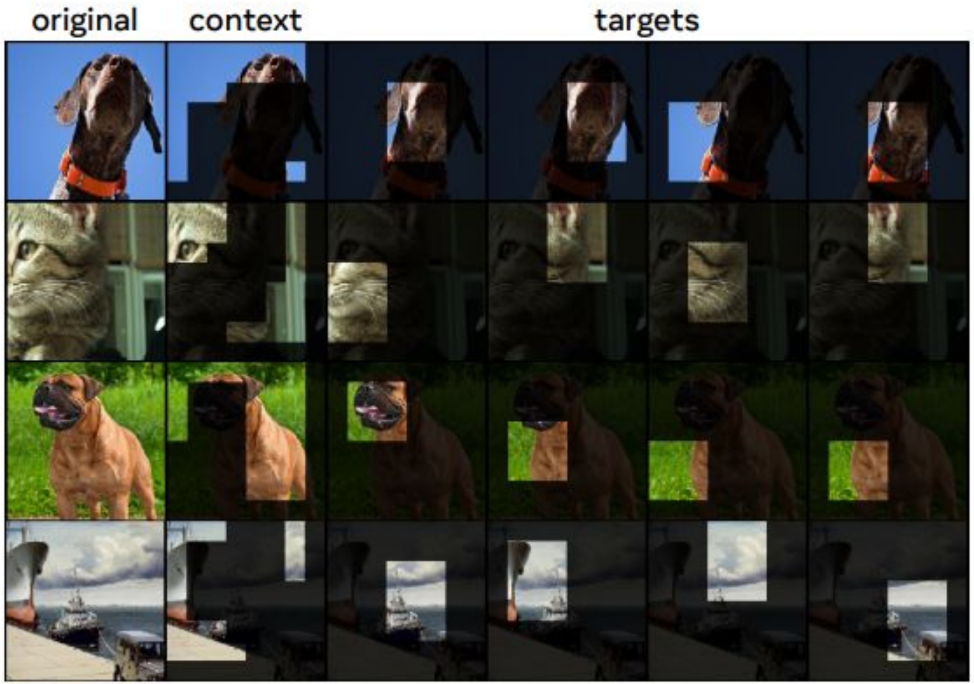
2. Input patches into target encoder to produce patch representations
   $\boldsymbol{s_y} = \{ \boldsymbol{s_{y1}}, \boldsymbol{s_{y2}}, \ldots, \boldsymbol{s_{yN}} \}$

3. Sample $M$ blocks from $\boldsymbol{s_y}$ randomly
   a. *B_i:* indices of patches in target block *i*
   b. Block *i*: $\boldsymbol{s_y}(i) = \{ \boldsymbol{s_{yj}} \}_{j \text{ in } B\_i}$

# I-JEPA Context Blocks

1. Randomly sample single context block from raw image ***x***
2. Remove overlapping regions between context and target blocks
3. Convert masked context block into non-overlapping patches
4. Input context block patches into context encoder to produce representations
   a. B_x: patch indices in context block
   b. Patch representations: $\boldsymbol{s}_x = \{ \boldsymbol{s}_{xj} \}_{j \text{ in } B\_x}$

# I-JEPA Target and Context Blocks



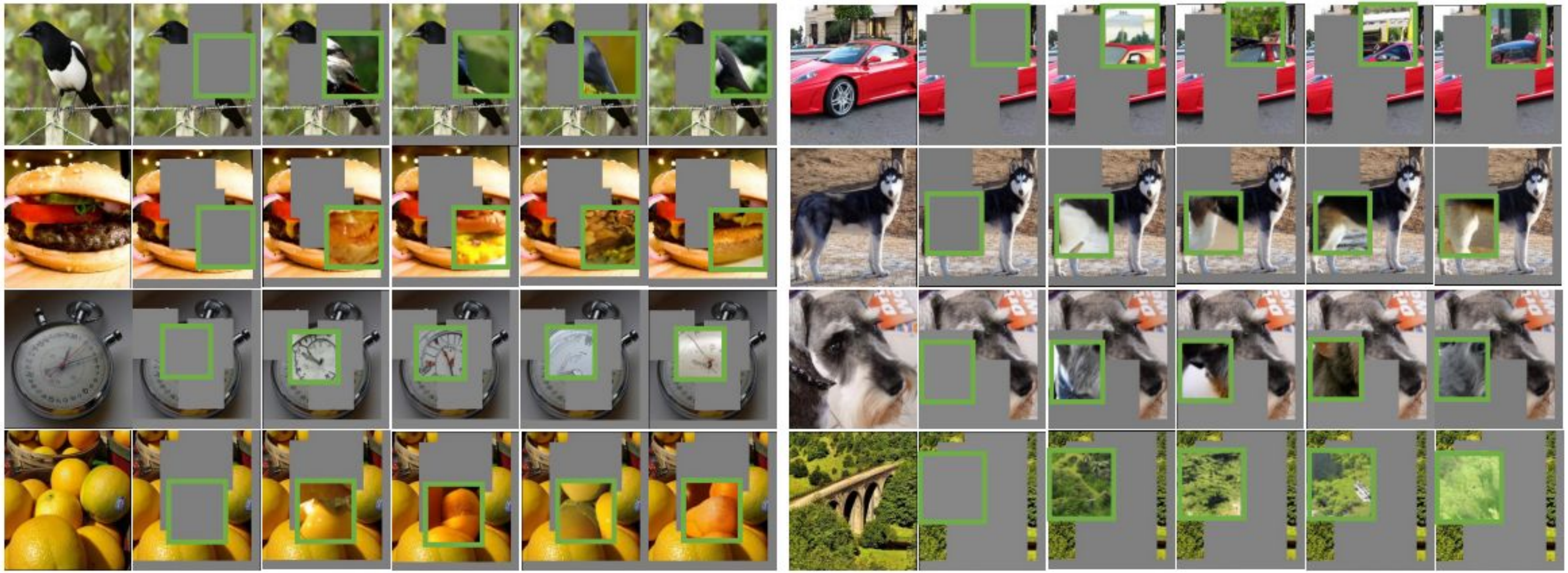
# I-JEPA Predictions

- Predictor ViT
  - Inputs: context block embedding, mask token of target block
  - Returns: patch embedding prediction
- Mask token: shared learnable vector and positional encoding
- Applied *M* times, one for each target block
  - Obtain *M* predictions $\boldsymbol{s\_hat}_{y}(1), \ldots, \boldsymbol{s\_hat}_{y}(M)$

- Loss function

$$\frac{1}{M}\sum_{i=1}^{M} D\left(\hat{\boldsymbol{s}}_y(i), \boldsymbol{s}_y(i)\right) = \frac{1}{M}\sum_{i=1}^{M}\sum_{j \in B_i} \|\hat{\boldsymbol{s}}_{y_j} - \boldsymbol{s}_{y_j}\|_2^2$$

# I-JEPA Prediction Visualization



# Experiments: Image Classification

- SSL pre-trained on ImageNet-1K
- ImageNet-1K linear-probing
  - Freeze model weights, train linear classifier on top on ImageNet-1K
- Few-Shot ImageNet-1K
  - Fine-tune or linear-probe on 1% of ImageNet labels
- Transfer Learning
  - Linear-probing on other image classification datasets

# Results: ImageNet-1K Linear-Probing

| Method | Arch. | Epochs | Top-1 |
|---|---|---|---|
| *Methods without view data augmentations* | | | |
| data2vec [8] | ViT-L/16 | 1600 | 77.3 |
| MAE [36] | ViT-B/16 | 1600 | 68.0 |
| | ViT-L/16 | 1600 | 76.0 |
| | ViT-H/14 | 1600 | 77.2 |
| CAE [22] | ViT-B/16 | 1600 | 70.4 |
| | ViT-L/16 | 1600 | 78.1 |
| I-JEPA | ViT-B/16 | 600 | 72.9 |
| | ViT-L/16 | 600 | 77.5 |
| | ViT-H/14 | 300 | 79.3 |
| | ViT-H/$16_{448}$ | 300 | **81.1** |
| *Methods using extra view data augmentations* | | | |
| SimCLR v2 [21] | RN152 (2×) | 800 | 79.1 |
| DINO [18] | ViT-B/8 | 300 | 80.1 |
| iBOT [79] | ViT-L/16 | 250 | **81.0** |

# Results: Few-Shot ImageNet-1K

| Method | Arch. | Epochs | Top-1 |
|---|---|---|---|
| *Methods without view data augmentations* | | | |
| data2vec [8] | ViT-L/16 | 1600 | 73.3 |
| MAE [36] | ViT-L/16 | 1600 | 67.1 |
| | ViT-H/14 | 1600 | 71.5 |
| I-JEPA | ViT-L/16 | 600 | 69.4 |
| | ViT-H/14 | 300 | 73.3 |
| | ViT-H/16$_{448}$ | 300 | **77.3** |
| *Methods using extra view data augmentations* | | | |
| iBOT [79] | ViT-B/16 | 400 | 69.7 |
| DINO [18] | ViT-B/8 | 300 | 70.0 |
| SimCLR v2 [35] | RN151 (2×) | 800 | 70.2 |
| BYOL [35] | RN200 (2×) | 800 | 71.2 |
| MSN [4] | ViT-B/4 | 300 | **75.7** |

# Results: Transfer Learning

| Method | Arch. | CIFAR100 | Places205 | iNat18 |
|---|---|---|---|---|
| *Methods without view data augmentations* | | | | |
| data2vec [8] | ViT-L/16 | 81.6 | 54.6 | 28.1 |
| MAE [36] | ViT-H/14 | 77.3 | 55.0 | 32.9 |
| I-JEPA | ViT-H/14 | **87.5** | **58.4** | **47.6** |
| *Methods using extra view data augmentations* | | | | |
| DINO [18] | ViT-B/8 | 84.9 | 57.9 | 55.9 |
| iBOT [79] | ViT-L/16 | **88.3** | **60.4** | **57.3** |

# Experiments: Low-Level Tasks

- SSL pre-trained on ImageNet-1K ?
- Object Counting
  - Linear-probing on Clevr/Count dataset
- Depth prediction
  - Linear-probing on Clevr/Dist dataset

# Results: Low-Level Tasks

| Method | Arch. | Clevr/Count | Clevr/Dist |
|---|---|---|---|
| *Methods without view data augmentations* | | | |
| data2vec [8] | ViT-L/16 | 85.3 | 71.3 |
| MAE [36] | ViT-H/14 | **90.5** | **72.4** |
| I-JEPA | ViT-H/14 | 86.7 | **72.4** |
| *Methods using extra data augmentations* | | | |
| DINO [18] | ViT-B/8 | 86.6 | 53.4 |
| iBOT [79] | ViT-L/16 | 85.7 | 62.8 |

# Ablation Studies

- Model and dataset scale
- Pixel vs. embedding space predictions
- Masking strategies
  - `multi-block`: proposed method
  - `rasterized`: split image into quadrants, and predict three quadrants using the fourth as context
  - `block`: target is single image block, context is image complement
  - `random`: target is set of random patches, context is image complement

# Ablation Studies: Scale

| Pretrain | Arch. | CIFAR100 | Place205 | INat18 | Clevr/Count | Clevr/Dist |
|---|---|---|---|---|---|---|
| IN1k | ViT-H/14 | 87.5 | 58.4 | 47.6 | 86.7 | 72.4 |
| IN22k | ViT-H/14 | **89.5** | 57.8 | 50.5 | **88.6** | **75.0** |
| IN22k | ViT-G/16 | **89.5** | **59.1** | **55.3** | 86.7 | 73.0 |

I-JEPA benefits from model and dataset size scale

# Ablation Studies: Components

| Targets | Arch. | Epochs | Top-1 |
|---|---|---|---|
| Target-Encoder Output | ViT-L/16 | 500 | **66.9** |
| Pixels | ViT-L/16 | 800 | 40.7 |

Best Few-Shot ImageNet-1K performance when SSL predictions made in embedding space

# Ablation Studies: Components

| | Targets | | Context | | |
|---|---|---|---|---|---|
| **Mask** | Type | Freq. | Type | Avg. Ratio* | **Top-1** |
| `multi-block` | Block(0.15, 0.2) | 4 | Block(0.85, 1.0) × Complement | 0.25 | **54.2** |
| `rasterized` | Quadrant | 3 | Complement | 0.25 | 15.5 |
| `block` | Block(0.6) | 1 | Complement | 0.4 | 20.2 |
| `random` | Random(0.6) | 1 | Complement | 0.4 | 17.6 |

**Avg. Ratio is the average number of patches in the context block relative to the total number of patches in the image.*

Best Few-Shot ImageNet-1K performance using
`multi-block` masking during SSL pre-training

# Conclusion

- I-JEPA: joint embedding-predictive SSL approach for images
  - No hand-crafted data augmentations
  - Generalizable to other modalities
- Predict *target blocks* based on single *context block*
  - Predictions made in *embedding space*
- **Claim:** can learn better semantic representations than other approaches that don't leverage data augmentations
  - **Personal opinion:** would have been better supported with attention map visuals
- Can close gap on, and even surpass, augmentation-based approaches

# Quiz

**Q:** What is the difference between joint-embedding and joint-embedding predictive SSL approaches?

# Quiz

**Q:** What is the difference between joint-embedding and joint-embedding predictive SSL approaches?

**A:** Joint-embedding SSL only aims to learn similar/dissimilar embeddings between similar/dissimilar inputs. Joint-embedding predictive SSL learns embeddings of a signal ***x*** and uses that, along with a conditional variable ***z***, to **predict** the (also learned) embeddings of another signal ***y***.

# Quiz

**Q:** How does I-JEPA choose the target and context blocks?

# Quiz

**Q:** How does I-JEPA choose the target and context blocks?

**A:** I-JEPA randomly samples *M* target blocks from the image patch embeddings. It then samples a single context block from the raw image, and then removes any overlapping regions between the context and target blocks.

# Thank you for listening!