We propose ADCLR: Accurate and Dense Contrastive Representation Learning, a novel self-supervised learning framework for learning accurate and dense vision representation. To extract spatial-sensitive information, ADCLR introduces query patches for contrasting in addition with global contrasting. Compared with previous dense contrasting methods, ADCLR mainly enjoys three merits: i) achieving both global-discriminative and spatial-sensitive representation, ii) model-efficient (no extra parameters in addition to the global contrasting baseline), and iii) correspondence-free and thus simpler to implement. Our approach achieves new state-of-the-art performance for contrastive methods. On classification tasks, for ViT-S, ADCLR achieves 77.5% top-1 accuracy on ImageNet with linear probing, outperforming our baseline (DINO) without our devised techniques as plug-in, by 0.5%. For ViT-B, ADCLR achieves 79.8%, 84.0% accuracy on ImageNet by linear probing and finetune, outperforming iBOT by 0.3%, 0.2% accuracy. For dense tasks, on MS-COCO, ADCLR achieves significant improvements of 44.3% AP on object detection, 39.7% AP on instance segmentation, outperforming previous SOTA method SelfPatch by 2.2% and 1.2%, respectively. On ADE20K, ADCLR outperforms SelfPatch by 1.0% mIoU, 1.2% mAcc on the segmentation task.
achieve state-of-the-art performance on downstream dense tasks. However, there are mainly three disadvantages in these dense contrasting methods. i) It is hard to balance the global and patch-level losses in the dense contrasting methods (Xie et al., 2021; Wang et al., 2021), causing their less competitive linear/fine-tune accuracy on the global classification task. ii) Establishing the correspondence among pixels/patches usually requires bilinear interpolation, which is complex and heavily sensitive to random crop augmentation (in an extreme case, if two views have no intersection parts, there’s no correspondence relation). iii) Each corresponding pixel/patch need be involved in the final contrastive loss, which is time-consuming. In this paper, we propose Accurate and Dense Contrastive Representation Learning (ADCLR), which is more global-discriminative, spatial-sensitive, correspondence-free and efficient. The main contributions of ADCLR are:

1) Cross-view Query-based Patch-level Contrasting Paradigm: For patch-level contrasting as recently used for dense tasks in negative-free Transformer-based SSL methods, we propose to augment two views, and perform contrasting crops from different views, for more effective learning with increased contrasting difficulty. The motivation for our cross-view design instead of the commonly-used single-view in existing patch-level contrasting is: it is non-trivial and even impossible to establish patch correspondence within a single view (especially adding random resized crop augmentation). While introducing two views and replacing the correspondence establishing with more feasible query could meanwhile increase the patch appearance variance for more difficult contrasting. The above module can be introduced to existing global contrasting only SSL baselines. As shown in Fig. 1, the [CLS] tokens are used to extract global information, and the designed query patches are used to extract local information, making ADCLR both global-discriminative and spatial-sensitive.

2) Robust Unidirectional Cross-Attention Scheme under the above Paradigm: The above patch-level contrasting paradigm can technically be prone to collapse to a trivial solution1 (see more explanation in our theoretical analysis in Sec. 3.3) if we directly resort to the unidirectional self-attention scheme as used in the vanilla vision Transformers. Instead, we design unidirectional cross-attention, which takes both query patches and raw patches from raw images as input. For attention block, the data flow are: $RP \rightarrow RP$, $\{RP, QP\} \rightarrow QP$, $QP \rightarrow QP$ ($i \neq j$) where $RP$ and $QP$ means raw patches (including [CLS] token) and the $i$-th query patch, respectively.

3) Boosting baselines to new SOTA accuracy on classification and dense tasks: The proposed ADCLR can serve as a plugin based on existing Transformer-based and negative-free SSL baselines e.g. DINO (Caron et al., 2021) and iBOT (Zhou et al., 2022). Our experimental results on both linear probing, fine-tune classification as well as other downstream tasks show the effectiveness of ADCLR.

2 RELATED WORKS

In this section, we review previous literature on general contrastive learning (for global tasks such as image classification) and dense ones (for downstream tasks like detection and segmentation).

General contrastive learning aims at learning the global information of images through enforcing contrastive objectives (e.g., InfoNCE loss, Hjelm et al. (2018)) on the final global image representations ([CLS] token or from average pooling). The pivotal idea is to align the embeddings of two augmented views from the same image while preventing trivial solutions (a.k.a. degenerated representations). To reach this target, MoCo (He et al., 2020; Chen et al., 2020b) employs a memory bank to store and update negative examples, from which the negative examples could be randomly sampled, whereas SimCLR (Chen et al., 2020a) treats all other data samples within the same training batch as negative examples. Due to the inconvenience and huge cost of using negative examples, later research turned to explore negative-example-free methods. BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2021) design asymmetric architectures with additional predictors and stop-gradient to prevent using negative examples. Barlow Twins (Zbontar et al., 2021), ZeroCL (Zhang et al., 2021) and VICReg (Bardes et al., 2022) resort to feature-level decorrelation to mitigate trivial solutions. Inspired by the success of Vision Transformers (ViT), CNN backbones are gradually replaced with ViTs Chen et al. (2021); Caron et al. (2021). iBOT (Zhou et al., 2022) further incorporates contrastive learning with masked image modeling and has achieved State-of-the-Art performance on various tasks.

Dense contrastive learning, on the contrary, targets learning local information through regularizing different local regions of images. One common dense contrastive learning way is to mine the correspondence of each pixel (CNNs) or patch (ViTs) in a feature map (Wang et al., 2021; Yun et al.,

---

1The embedding vectors end up spanning a lower-dimensional subspace instead of the entire space.
We first review recent SSL approaches with ViTs as backbones (Caron et al., 2021; Yun et al., 2022) where the input embeddings can be formulated as:

\[ \mathbf{z} = \left[ \mathbf{z}^{[CLS]}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(N)} \right] \]

where each query patch into a linear projector to get its embedding as raw patches (e.g., \([16 \times 16 \times 8 \times 8]\)). To this end, we randomly crop each image to generate \(Q\) query crops. Although DINO is able to handle most cases (e.g., two augmented views contain the same objects), we find that the \([CLS]\) token may extract mismatched information (see the attention map of \([CLS]\) in Fig. 1). To this end, we randomly crop each image to generate \(Q\) query crops, where \(Q\) is a pre-defined hyper-parameter. Then, we resize each query crop to the same resolution as raw patches (e.g., \([16 \times 16 \times 8 \times 8]\)): \(\mathbf{X}^q \in \mathbb{R}^{Q \times CP^2}\). Denote the \(i\)-th query crop as \(\mathbf{x}^q_i\), we feed each query patch into a linear projector to get its embedding \(\mathbf{z}^{(q_i)}\). Different from raw patches, it’s unnecessary to add positional embeddings to query patches as the region of the query crop may not in the augmented views. Then, the input embeddings can be formulated as:

\[ \mathbf{z} = \left[ \mathbf{z}^{[CLS]}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(N)} \right] \]

\[ = \left[ \mathbf{z}^{(q_1)}, \mathbf{z}^{(q_2)}, \ldots, \mathbf{z}^{(q_Q)} \right] \]

where \(\mathbf{z}^{(q_i)}\) is the \(i\)-th patch, respectively. A \([CLS]\) token may extract mismatched information (see the attention map of \([CLS]\) in Fig. 1).

**3 Methodology**

We first review recent SSL approaches with ViTs as backbones (Caron et al., 2021; Yun et al., 2022) in Sec. 3.1. Then we present ADCLR detailedly in Sec. 3.2 (we provide a sketch in Fig. 1).

### 3.1 Preliminaries

**Vision Transformers.** Denote an image by \(\mathbf{x} \in \mathbb{R}^{C \times H \times W}\), where \(H \times W\) is the resolution of the image and \(C\) is the number of channels. Plain ViT (Dosovitskiy et al., 2020) treats the image \(\mathbf{x}\) as a sequence composed of non-overlapping patches \(\{\mathbf{x}^{(i)} \in \mathbb{R}^{CP^2}\}_{i=1}^N\), where each patch has a fixed \(P \times P\) resolution. Then, the patches are linearly transformed to \(D\)-dimensional patch embeddings \(\mathbf{z}^{(i)} = \mathbf{E}\mathbf{x}^{(i)} + \mathbf{W}_{pos}^{(i)} \in \mathbb{R}^{D}\), where \(\mathbf{E} \in \mathbb{R}^{D \times CP^2}\) is the linear projection and \(\mathbf{W}_{pos} \in \mathbb{R}^D\) is the positional embedding for the \(i\)-th patch. A \([CLS]\) token \(\mathbf{z}^{[CLS]} \in \mathbb{R}^D\) is subsequently prepended to the patch sequence to extract global information, so the resulting input sequence is represented as \(\mathbf{z} = [\mathbf{z}^{[CLS]}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(N)}]\). Then, ViT uses a Transformer encoder (Vaswani et al., 2017) to generate both image-level ([CLS] token) and patch-level (other tokens). In line with SelfPatch (Yun et al., 2022), we use \(f_\theta\) to denote the whole process of a ViT parameterized by \(\theta\):

\[
f_\theta(\mathbf{x}) = f_\theta\left(\left[\mathbf{z}^{[CLS]}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(N)}\right]\right) = \left[ f^{[CLS]}_\theta(\mathbf{x}), f^{(1)}_\theta(\mathbf{x}), f^{(2)}_\theta(\mathbf{x}), \ldots, f^{(N)}_\theta(\mathbf{x}) \right],
\]

(1)

where \(f^{[CLS]}_\theta(\mathbf{x})\) and \(f^{(i)}_\theta(\mathbf{x})\) are the representations of the whole image and \(i\)-th patch, respectively.

**Self-supervised learning with ViTs.** Since our ADCLR is built on top of DINO and iBOT, we shortly review DINO’s framework and objective function. Given the image \(\mathbf{x}\), DINO constructs a positive pair \((\mathbf{x}_A, \mathbf{x}_B)\) through random augmentation. Then, DINO learns by maximizing the similarity between their representations. The generic form of image-level self-supervised loss is:

\[
\mathcal{L}_{DINO} = H \left( g_{\gamma} \left( f^{[CLS]}_\theta(\mathbf{x}_A) \right), sg \left( g_{\gamma} \left( f^{[CLS]}_\theta(\mathbf{x}_B) \right) \right) \right),
\]

(2)

where \(H(a, b) = -a \log b\) is the cross entropy loss. \(sg(\cdot)\) means stop-gradient operation. \(g_\gamma\) is the MLP projector, which is commonly used in previous SSL methods (Chen et al., 2020a; Grill et al., 2020). \(\gamma\) and \(\theta\) denote the exponential moving averages updated parameters in the teacher branch.

### 3.2 ADCLR: Accurate and Dense Contrastive Representation Learning

**Query crops.** Although DINO is able to handle most cases (e.g., two augmented views contain the same objects), we find that the \([CLS]\) token may extract mismatched information (see the attention map of \([CLS]\) in Fig. 1). To this end, we randomly crop each image to generate \(Q\) query crops, where \(Q\) is a pre-defined hyper-parameter. Then, we resize each query crop to the same resolution as raw patches (e.g., \(16 \times 16\) or \(8 \times 8\)): \(\mathbf{X}^q \in \mathbb{R}^{Q \times CP^2}\). Denote the \(i\)-th query crop as \(\mathbf{x}^q_i\), we feed each query patch into a linear projector to get its embedding \(\mathbf{z}^{(q_i)}\). Different from raw patches, it’s unnecessary to add positional embeddings to query patches as the region of the query crop may not in the augmented views. Then, the input embeddings can be formulated as:

\[ \mathbf{z} = \left[ \mathbf{z}^{[CLS]}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(N)}, \mathbf{z}^{(q_1)}, \mathbf{z}^{(q_2)}, \ldots, \mathbf{z}^{(q_Q)} \right] \]

(3)
Augustina
Augmentation
Query
Crop
Transformer Encoder
EMA update
CLS
Token
Bidirectional self-attention in standard vision Transformer leads to collapse. In a nutshell, we can summarize the propagation rule as easily to collapse if

\[ Q \text{ is large enough (} Q >> \text{ the total number of raw patches of the image).} \]

Note that \( z^{CLS} \) and raw patches \( \{ z^{(1)}, z^{(2)}, \ldots, z^{(N)} \} \) will be equipped with positional embeddings.

**Unidirectional cross attention mechanism.** Instead of directly feeding the whole sequence (namely \( CLS \) token, raw patches, query patches) to the Transformer encoder, which may hurt the performance of the downstream task (as only \( CLS \) token and raw patches are fed to the Transformer for downstream tasks), we slightly modify the attention mechanism. Given one \( z^{CLS} \) token, raw patches sequence \( \{ z^{(1)}, z^{(2)}, \ldots, z^{(N)} \} \) and query patches sequence \( \{ z^{(q_1)}, z^{(q_2)}, \ldots, z^{(q_Q)} \} \). Let \( z' = [z^{CLS}, z^{(1)}, z^{(2)}, \ldots, z^{(N)}] \) and \( z'' = [z^{(q_1)}, z^{(q_2)}, \ldots, z^{(q_Q)}] \). For \( z^{CLS} \) and raw patches, the propagation rule in the attention block is in line with plain Transformer, i.e.,

\[
\text{Attn}(Q_{z'}, K_{z'}, V_{z'}) = \text{Softmax} \left( \frac{Q_{z'} K_{z'}^T}{\sqrt{d_k}} \right) V_{z'}, \tag{4}
\]

where \( Q_{z'} = z' W_Q, K = z' W_K, V = z' W_V, \) and \( W_Q, W_K, W_V \) are learnable weights. For query patches \( z^{(q_i)} \), the attention propagation rule is similar to \( CLS \) token, i.e.,

\[
\text{Attn}(Q_{z^{(q_i)}}, K_{z'}, V_{z'}) = \text{Softmax} \left( \frac{Q_{z^{(q_i)}} K_{z'}^T}{\sqrt{d_k}} \right) V_{z'}, \quad 1 \leq i \leq Q, \tag{5}
\]

where \( Q_{z^{(q_i)}} = z^{(q_i)} W_Q \). By Eq. 4 and Eq. 5, we can complete the attention block in modified Transformer. In a nutshell, we can summarize the propagation rule as \( RP \rightarrow RP, \{ RP, QP_i \} \rightarrow QP_j, QP_i \rightarrow QP_j \) (\( i \neq j \)), where \( RP \) and \( QP \) means raw patches (including \( CLS \) token) and \( i \)-th query patch, respectively. We can find only raw patches and \( CLS \) token linearly express the raw patches and \( CLS \) in the attention mechanism. This design can make the pretrained Transformer more suitable for downstream tasks without introducing extra parameters.

**Objective function of ADCLR.** By feeding the whole sequence of two views \( x_A \) and \( x_B \) to the modified Transformer \( f_B \) and MLP projection head \( g_\gamma \), we can obtain the final representation of the image \( h^{CLS}_{A,B} \), raw patches \( h_{A:B} \) and query patches \( h_{A:B}^{q} \). Then, the overall objectives are:

\[
\mathcal{L}_{ADCLR} = H(h_A^{CLS}, h_B^{CLS}) + \frac{\lambda}{Q} \sum_{i=1}^{Q} H(h_A^{(q_i)}, h_B^{(q_i)}), \tag{6}
\]

where \( \lambda \) is used to balance global and local loss. The impact of \( \lambda \) is studied in Sec. 4.3. ADCLR can also selectively add the MIM objective in iBOT Zhou et al. (2022) to further boost the accuracy.

### 3.3 Necessity of Our Unidirectional Cross-Attention Mechanism

**Theorem 1** Bidirectional self-attention in standard vision Transformer leads to collapse. Given a set of query patches \( \{ x_i \}^Q_{i=1} \) and a set of raw patches \( \{ x_i \}^N_{i=1} \), contrasting with the standard vision Transformer-like backbones that use bidirectional self-attention mechanisms can cause ADCLR easily to collapse if \( Q \) is large enough (\( Q >> \) the total number of raw patches of the image).
Table 1: Linear and finetune Top-1 accuracy on ImageNet-1K. † means using DALL-E (Reddy et al., 2021) tokenizer in the pre-training stage. ‡ means these works are so far arXiv preprints to date. * means adding the mask image modeling loss proposed in (Zhou et al., 2022)

<table>
<thead>
<tr>
<th>Framework</th>
<th>Method</th>
<th>Model</th>
<th>#Params</th>
<th>PT Eps.</th>
<th>Linear</th>
<th>FT Eps.</th>
<th>FT Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training from scratch</td>
<td>Scratch, DeiT</td>
<td>ViT-B</td>
<td>86M</td>
<td>0</td>
<td>300</td>
<td>81.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scratch, MAE</td>
<td>ViT-B</td>
<td>86M</td>
<td>0</td>
<td>300</td>
<td>82.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scratch, Swin</td>
<td>Swin-B</td>
<td>88M</td>
<td>0</td>
<td>300</td>
<td>83.5</td>
<td></td>
</tr>
<tr>
<td>Supervised Pre-training</td>
<td>Supervised, SimMIM</td>
<td>Swin-B</td>
<td>88M</td>
<td>300</td>
<td>100</td>
<td>83.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supervised, SimMIM</td>
<td>Swin-L</td>
<td>197M</td>
<td>300</td>
<td>100</td>
<td>83.5</td>
<td></td>
</tr>
<tr>
<td>Masked Image Modeling</td>
<td>MAE</td>
<td>ViT-B</td>
<td>86M</td>
<td>1600</td>
<td>67.8</td>
<td>100</td>
<td>83.6</td>
</tr>
<tr>
<td></td>
<td>BEiT†</td>
<td>ViT-B</td>
<td>86M</td>
<td>300</td>
<td>37.6</td>
<td>100</td>
<td>83.0</td>
</tr>
<tr>
<td></td>
<td>CAE†‡</td>
<td>ViT-B</td>
<td>86M</td>
<td>300</td>
<td>64.2</td>
<td>100</td>
<td>83.3</td>
</tr>
<tr>
<td></td>
<td>CAE†‡</td>
<td>ViT-B</td>
<td>86M</td>
<td>800</td>
<td>68.3</td>
<td>100</td>
<td>83.6</td>
</tr>
<tr>
<td></td>
<td>CIM‡</td>
<td>ViT-B</td>
<td>86M</td>
<td>300</td>
<td>-</td>
<td>100</td>
<td>83.3</td>
</tr>
<tr>
<td>Contrastive Learning</td>
<td>Moco V3</td>
<td>ViT-B</td>
<td>86M</td>
<td>800</td>
<td>76.5</td>
<td>100</td>
<td>83.2</td>
</tr>
<tr>
<td></td>
<td>DINO</td>
<td>ViT-B</td>
<td>86M</td>
<td>400</td>
<td>78.2</td>
<td>100</td>
<td>83.6</td>
</tr>
<tr>
<td></td>
<td>iBOT</td>
<td>ViT-B</td>
<td>86M</td>
<td>400</td>
<td>79.5</td>
<td>100</td>
<td>83.8</td>
</tr>
<tr>
<td></td>
<td>ADCLR</td>
<td>ViT-B</td>
<td>86M</td>
<td>400</td>
<td><strong>78.6</strong></td>
<td>100</td>
<td><strong>83.9</strong></td>
</tr>
<tr>
<td></td>
<td>ADCLR*</td>
<td>ViT-B</td>
<td>86M</td>
<td>400</td>
<td><strong>79.8</strong></td>
<td>100</td>
<td><strong>84.0</strong></td>
</tr>
</tbody>
</table>

Remarks. In fact, applying the standard bidirectional self-attention as used in vision Transformers not only incurs the solution collapse issue as shown in Theorem 1, but also may hurt the performance, as will be shown in the ablation studies in Sec. 4.3. We guess this is because ADCLR introduces additional query patches during the pretraining stage, while the downstream task finetuning only involves raw patches and the bidirectional scheme would cause the mutual influence between query and raw patches, leading to the input inconsistency between pretraining and downstream tasks. Therefore, we elaborately replace the bidirectional self-attention scheme by our devised unidirectional cross-attention one as such the above two issues are addressed directly.

4 EXPERIMENTS

4.1 Experiment Setup

Dataset. We conduct self-supervised pre-training on the ImageNet-1K (Deng et al., 2009) training set, as commonly used in SSL methods for both MIM (He et al., 2021) and contrastive learning (Chen et al., 2020a). ImageNet-1k includes 1,000 classes which are even in distribution. We also transfer the encoder pretrained by ADCLR on MS-COCO (Lin et al., 2014), ADE20K (Zhou et al., 2017) and other fine-grained classification datasets (Van Horn et al., 2018) for downstream evaluation.

Baselines. We consider a variety of existing SSL methods based on the ResNet (He et al., 2016) and ViT (Dosovitskiy et al., 2020) architectures: (a) self-supervised ResNets: MoCo-v2 (Chen et al., 2020b), SwAV (Caron et al., 2020), Barlow Twins (Zbontar et al., 2021), ZeroCL (Zhang et al., 2021), ARB (Zhang et al., 2022), DenseCL (Wang et al., 2021), ReSim (Xiao et al., 2021), and DetCo (Xie et al., 2021a); and (b) self-supervised ViTs: DINO (Caron et al., 2021), MoCo-v3 (Chen et al., 2021), MoBY (Xie et al., 2021b), iBOT (Zhou et al., 2022) and SelfPatch (Yun et al., 2022).

Evaluation protocols. Standard SSL protocols is to either learn a linear classifier on frozen features (Chen et al., 2020a; He et al., 2020) or to finetune on downstream tasks (He et al., 2021; Chen et al., 2022). For linear evaluations, we apply random resize crops and horizontal flips augmentation for training, and report accuracy on a central crop. For finetuning evaluations (detection and segmentation on MS-COCO (Lin et al., 2014), segmentation on ADE20K (Zhou et al., 2017)), we initialize networks with the pretrained weights to adapt with further training. In line with (Zbontar et al., 2021), we also evaluate our method’s transfer ability on small-scale and fine-grained classification dataset.

4.2 Main Results

We first validate the ADCLR framework used in this study with the standard self-supervised benchmark on ImageNet. Then, we study the performance of detection, segmentation, and transfer learning.

Finetune on ImageNet-1k. Table 1 shows the results with four different learning schemes, i.e., finetune from scratch, supervised pre-training, MIM-based approaches, and contrastive learning. We report linear probing and finetune accuracy, respectively. Our finetune protocol is in line with iBOT (Zhou et al., 2022), using strong regularization for training 100 epochs. The compared baselines
Table 2: $k$-NN and linear top-1 accuracy on ImageNet-1k. Epoch$^1$ means effective pre-training epochs accounting for actual trained images/views, which is used in iBOT (Zhou et al., 2022). * means using MIM loss and #Views means the number of views used in the pretraining stage.

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>#Params</th>
<th>images/s</th>
<th>#Views</th>
<th>Epoch$^1$</th>
<th>$k$-NN</th>
<th>Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>RN50</td>
<td>23M</td>
<td>1237</td>
<td>1</td>
<td>300</td>
<td>79.3</td>
<td>79.3</td>
</tr>
<tr>
<td>SinCL (Chen et al., 2020a)</td>
<td>RN50</td>
<td>23M</td>
<td>1237</td>
<td>2</td>
<td>1000</td>
<td>60.1</td>
<td>69.1</td>
</tr>
<tr>
<td>MoCoV2 (Chen et al., 2020b)</td>
<td>RN50</td>
<td>23M</td>
<td>1237</td>
<td>2</td>
<td>800</td>
<td>61.9</td>
<td>71.1</td>
</tr>
<tr>
<td>ARB (Zhang et al., 2022)</td>
<td>RN50</td>
<td>23M</td>
<td>1237</td>
<td>2</td>
<td>1000</td>
<td>66.4</td>
<td>73.1</td>
</tr>
<tr>
<td>BYOL (Grill et al., 2020)</td>
<td>RN50</td>
<td>23M</td>
<td>1237</td>
<td>2</td>
<td>1000</td>
<td>66.0</td>
<td>73.2</td>
</tr>
<tr>
<td>SwAV (Caron et al., 2020)</td>
<td>RN50</td>
<td>23M</td>
<td>1237</td>
<td>8</td>
<td>800</td>
<td>65.7</td>
<td>75.3</td>
</tr>
<tr>
<td>DINO (Caron et al., 2021)</td>
<td>RN50</td>
<td>23M</td>
<td>1237</td>
<td>12</td>
<td>3200</td>
<td>67.5</td>
<td>75.3</td>
</tr>
<tr>
<td>Moco V3 (Chen et al., 2021)</td>
<td>ViT-S/16</td>
<td>22M</td>
<td>1007</td>
<td>2</td>
<td>1200</td>
<td>73.4</td>
<td></td>
</tr>
<tr>
<td>Moco V3 (Chen et al., 2021)</td>
<td>ViT-B/16</td>
<td>66M</td>
<td>312</td>
<td>2</td>
<td>1200</td>
<td>76.7</td>
<td></td>
</tr>
<tr>
<td>SwAV (Caron et al., 2020)</td>
<td>ViT-S/16</td>
<td>22M</td>
<td>1007</td>
<td>8</td>
<td>2400</td>
<td>66.3</td>
<td>73.5</td>
</tr>
<tr>
<td>DINO (Caron et al., 2021)</td>
<td>ViT-S/16</td>
<td>22M</td>
<td>1007</td>
<td>12</td>
<td>3200</td>
<td>74.5</td>
<td>77.0</td>
</tr>
<tr>
<td>DINO (Caron et al., 2021)</td>
<td>ViT-B/16</td>
<td>66M</td>
<td>312</td>
<td>12</td>
<td>1600</td>
<td>76.1</td>
<td>78.2</td>
</tr>
<tr>
<td>iBOT (Zhou et al., 2022)</td>
<td>ViT-S/16</td>
<td>22M</td>
<td>1007</td>
<td>12</td>
<td>3200</td>
<td>75.2</td>
<td>77.9</td>
</tr>
<tr>
<td>iBOT (Zhou et al., 2022)</td>
<td>ViT-B/16</td>
<td>66M</td>
<td>312</td>
<td>12</td>
<td>1600</td>
<td>77.1</td>
<td>79.5</td>
</tr>
<tr>
<td>ADCLR</td>
<td>ViT-S/16</td>
<td>22M</td>
<td>1007</td>
<td>12</td>
<td>3200</td>
<td>74.6</td>
<td>77.5</td>
</tr>
<tr>
<td>ADCLR</td>
<td>ViT-B/16</td>
<td>66M</td>
<td>312</td>
<td>12</td>
<td>1600</td>
<td>76.6</td>
<td>78.6</td>
</tr>
<tr>
<td>ADCLR*</td>
<td>ViT-S/16</td>
<td>22M</td>
<td>1007</td>
<td>12</td>
<td>3200</td>
<td>74.9</td>
<td>78.1</td>
</tr>
<tr>
<td>ADCLR*</td>
<td>ViT-B/16</td>
<td>66M</td>
<td>312</td>
<td>12</td>
<td>1600</td>
<td>77.4</td>
<td>79.8</td>
</tr>
</tbody>
</table>

Table 3: Object detection and instance segmentation on MS-COCO. Mask R-CNN is adopted and trained with the 1x schedule. All the results are based on the same implementation for object detection and instance segmentation. Epoch refers to the number of pretraining epochs on ImageNet-1K.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Epoch</th>
<th>#Param.</th>
<th>Detection</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AP$^b_50$</td>
<td>AP$_{50}^{mk}$</td>
</tr>
<tr>
<td>Moco-V2 (Chen et al., 2020b)</td>
<td>RN50</td>
<td>200</td>
<td>23M</td>
<td>38.9</td>
<td>59.2</td>
</tr>
<tr>
<td>SwAV (Caron et al., 2020)</td>
<td>RN50</td>
<td>200</td>
<td>23M</td>
<td>38.5</td>
<td>60.4</td>
</tr>
<tr>
<td>DenseCL (Wang et al., 2021)</td>
<td>RN50</td>
<td>200</td>
<td>23M</td>
<td>40.3</td>
<td>59.9</td>
</tr>
<tr>
<td>ReSim (Xiao et al., 2021)</td>
<td>RN50</td>
<td>200</td>
<td>23M</td>
<td>40.3</td>
<td>60.6</td>
</tr>
<tr>
<td>DetCo (Xie et al., 2021a)</td>
<td>RN50</td>
<td>200</td>
<td>23M</td>
<td>40.1</td>
<td>61.0</td>
</tr>
<tr>
<td>Moco V3 (Chen et al., 2021)</td>
<td>ViT-S/16</td>
<td>300</td>
<td>23M</td>
<td>39.8</td>
<td>62.6</td>
</tr>
<tr>
<td>MoBY (Xie et al., 2021b)</td>
<td>ViT-S/16</td>
<td>300</td>
<td>22M</td>
<td>41.1</td>
<td>63.7</td>
</tr>
<tr>
<td>DINO (Caron et al., 2021)</td>
<td>ViT-S/16</td>
<td>300</td>
<td>22M</td>
<td>40.8</td>
<td>63.4</td>
</tr>
<tr>
<td>SelfPatch (Yun et al., 2022)</td>
<td>ViT-S/16</td>
<td>300</td>
<td>22M</td>
<td>42.1</td>
<td>64.9</td>
</tr>
<tr>
<td>ADCLR</td>
<td>ViT-S/16</td>
<td>200</td>
<td>22M</td>
<td>43.8</td>
<td>65.2</td>
</tr>
<tr>
<td>ADCLR</td>
<td>ViT-S/16</td>
<td>300</td>
<td>22M</td>
<td>44.3</td>
<td>65.4</td>
</tr>
</tbody>
</table>

are MIM-based methods MAE (He et al., 2021), BEiT (Bao et al., 2021), CIM (Fang et al., 2022) as well as contrastive methods (Caron et al., 2021; Zhou et al., 2022; Chen et al., 2021) and the combination of the two techniques: CAE (Chen et al., 2022) which is emerging.

**Linear probing on ImageNet-1k.** For linear probing evaluation, we compare ADCLR with both CNNs-based and ViTs-based contrastive learning methods. Following iBOT (Zhou et al., 2022), we use effective training epochs to measure the images actually seen by ADCLR. Specifically, DINO and iBOT are by default trained with 2 global crops with the crop size 224*224 and 10 local crops with the size 96*96. The calculation formula is $r = 2 + (\frac{96}{224})^2 \times 2 = 3.84 \approx 4$. Moreover, ADCLR introduces query crops, and the total seen image is $r = 2 + (\frac{96}{224})^2 \times 10 + (\frac{48}{224})^2 \times 10 \approx 3.89 \approx 4$. The overall seen image brought by query crops is only 0.05, which can be negligible. We report $k$-NN and linear probing accuracy in Table 2. ADCLR achieves 77.5% top-1 accuracy on ImageNet-1k with ViT-S, outperforming baseline DINO +0.5% in 3200 effective epochs. For ViT-B, ADCLR outperforms DINO and iBOT +0.4% and +0.3% accuracy in 800 effective epochs, respectively.

### 4.2.1 Transfer Learning across Different Tasks

**MS-COCO Setup.** We evaluate pre-trained models on the MS-COCO object detection and instance segmentation tasks. Here, all models are fine-tuned with Mask R-CNN (He et al., 2017) and FPN (Lin et al., 2017) under the standard 1x schedule. Following (Chen et al., 2022), we utilize multi-scale training and resize the image with the size of the short side between 480 and 800 and the long side no larger than 1333. In pretraining stage, we set $\lambda = 0.5$ and $Q = 10$ for extract better dense information. The pretraining is in line with SelfPatch (Yun et al., 2022). In finetune stage, we distributed training...
on 8 Tesla V100 32G GPUs set batch size 32 and set the learning rate as 3e-4 with the layer decay rate 0.75, which also follows SelfPatch.

**Results.** Table 3 shows the detection and segmentation performance on MS-COCO. We give the results of both 200 and 300 epochs’ pretraining to better demonstrate the advance of ADCLR. We find that although both SelfPatch and ADCLR perform patch-level contrastive loss, ADCLR breaks the limitation of contrasting between the same view. The learning paradigm of query tokens seems to bring more variance than single view contrasting, increasing the difficulty of patch-level pretext tasks, and resulting in higher performance on detection and segmentation tasks. Specifically, under 200 epochs pretraining, ADCLR outperforms SelfPatch with +1.7% and +0.7% points on detection (\(\text{AP}^{bb}\)) and segmentation (\(\text{AP}^{mk}\)) tasks, respectively. Compared with baseline DINO (Caron et al., 2021), ADCLR outperforms DINO with a large range, e.g., +2.2% bounding box average precision, +1.2% mask average precision higher than DINO, which benefits from contrasting on query patches.

Table 4: **ADE20K semantic segmentation** performances of the recent self-supervised approaches pre-trained on ImageNet. The metrics mIoU, aAcc, and mAcc denote the mean intersection of union, all pixel accuracy, and mean class accuracy, respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Arch</th>
<th>Backbone</th>
<th>#Iter.</th>
<th>mIoU</th>
<th>aAcc</th>
<th>mAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo-v2 (Chen et al., 2020b)</td>
<td>FPN</td>
<td>ResNet50</td>
<td>40k</td>
<td>35.8</td>
<td>77.6</td>
<td>45.1</td>
</tr>
<tr>
<td>SwAV (Caron et al., 2020)</td>
<td>FPN</td>
<td>ResNet50</td>
<td>40k</td>
<td>35.4</td>
<td>77.5</td>
<td>44.9</td>
</tr>
<tr>
<td>DenseCL (Wang et al., 2021)</td>
<td>FPN</td>
<td>ResNet50</td>
<td>40k</td>
<td>37.2</td>
<td>78.5</td>
<td>47.1</td>
</tr>
<tr>
<td>MoBY (Xie et al., 2021b)</td>
<td>FPN</td>
<td>ViT-S/16</td>
<td>40k</td>
<td>35.3</td>
<td>78.9</td>
<td>45.9</td>
</tr>
<tr>
<td>DINO (Caron et al., 2021)</td>
<td>FPN</td>
<td>ViT-S/16</td>
<td>40k</td>
<td>39.5</td>
<td>79.9</td>
<td>50.5</td>
</tr>
<tr>
<td>DINO (Caron et al., 2021)</td>
<td>FPN</td>
<td>ViT-S/16</td>
<td>40k</td>
<td>38.3</td>
<td>79.0</td>
<td>49.4</td>
</tr>
<tr>
<td>SelfPatch (Yun et al., 2022)</td>
<td>FPN</td>
<td>ViT-S/16</td>
<td>40k</td>
<td>41.2</td>
<td>80.7</td>
<td>52.1</td>
</tr>
<tr>
<td>SelfPatch (Yun et al., 2022)</td>
<td>FPN</td>
<td>ViT-S/16</td>
<td>160k</td>
<td>43.2</td>
<td>81.5</td>
<td>53.9</td>
</tr>
<tr>
<td>ADCLR</td>
<td>FPN</td>
<td>ViT-S/16</td>
<td>40k</td>
<td>42.4</td>
<td>81.1</td>
<td>54.2</td>
</tr>
<tr>
<td>ADCLR</td>
<td>FPN</td>
<td>ViT-S/16</td>
<td>160k</td>
<td>44.2</td>
<td>81.8</td>
<td>55.1</td>
</tr>
</tbody>
</table>

Semantic segmentation on ADE20K. **Setups.** We also evaluate semantic segmentation performances of pre-trained models on ADE20K (Zhou et al., 2017), which includes 150 fine-grained semantic categories and 25k training data. In line with SelfPatch (Yun et al., 2022), we report three metrics: the mean intersection of union (mIoU) averaged over all semantic categories, all pixel accuracy (aAcc), and mean class accuracy (mAcc). We pretrain ADCLR in 200 epochs and finetune the pretrained backbone in different architectures (FPN (Lin et al., 2017) and UperNet (Xiao et al., 2018)) for fair comparisons. **Results.** The results are given in Table 4. When finetuning the FPN with 40k iterations, ADCLR outperforms SelfPatch and DINO +1.2% and +3.9% mIoU points, respectively. In UperNet with 160k iterations, ADCLR outperforms SelfPatch and DINO +1.0% and +3.0% mIoU points. We guess the gain mainly comes from two aspects, 1) patch-level contrasting enhances the ability to capture local information (see the gap between global method DINO and local method SelfPatch and ADCLR); 2) The learning paradigm of query tokens brings more accurate information (the gap between SelfPatch and ADCLR).

**Transfer learning on the small-scale dataset.** We study transfer learning where we pre-train on ImageNet-1K and fine-tune on several smaller datasets. We follow the training recipe and protocol used in DINO and iBOT. For pretraining, ViT-S and ViT-B are pretrained on 16 GPUs with 1024 batch size and 10 local views in 800 and 400 epochs, respectively. The results are reported in Table 5. While the results on several datasets (e.g., CIFAR10, CIFAR100, Flowers, and Cars) have almost plateaued, ADCLR consistently performs favorably against other SSL frameworks, achieving state-of-the-art transfer results. Specifically, on small-scale datasets, ADCLR outperforms baseline DINO (Caron et al., 2021) over 2.2% and 1.1% accuracy on iNaturalist18 and iNaturalist19 datasets. We observe ADCLR achieve greater performance gain over DINO in fine-grained classification datasets like iNaturalist18, iNaturalist19, and Flowers than CIFAR10 and CIFAR100, indicating ADCLR (query token) is more adept at capturing local information. Compared with the state-of-the-art method iBOT (Zhou et al., 2022), although ADCLR achieves similar results, iBOT spends more memory due to the MIM branch (under the same setting, 19.5G for iBOT and 16.7G for ADCLR).

### 4.3 Ablation Study

To verify the robustness of ADCLR, we comprehensively study the sensitiveness of balance ratio \(\lambda\), number of query tokens \(Q\), the effect of cross attention mechanism, and crop ratio.
Effect of \( \lambda \). We conduct ablation experiments on different balance ratios \( \lambda \) to explore the effect of local objective \( L_{\text{Local}} \). Specifically, we set batch size as 1024, and distribute training on 32 GPUs with 30 epochs as warm up out of the total 100 epochs. For efficiency, we only use two global views (224 * 224), and 10 query views (16 * 16) without local view (96 * 96). We report the results in Fig. 2(a). Compared with baseline DINO, ADCLR outperforms +1.3% top-1 accuracy when \( \lambda = 0.5 \). With too large \( \lambda \), the accuracy will drop a little, which we guess ADCLR would overemphasize the local information, but ignore the global information and leading to the decreasing accuracy of global task (classification). With smaller \( \lambda \) e.g. between 0.1 and 0.2, ADCLR is mainly determined by [CLS] token, which may bring unwanted contrasting loss (see Fig. 1) to hurt the final performance.

Effect of the number of query crops \( Q \). We set batch size as 1024 and pretrain ADCLR with 100 epochs. In line with ablation on \( \lambda \), we only use two global views without local views. Besides, we choose the best value of \( \lambda = 0.5 \) for this experiment. We illustrate the results in Fig. 2(b). The performance increases with the increase of \( Q \), which is similar to the ensemble mechanism (with more base models, the ensemble model are more accurate). We find ADCLR saturates when \( Q = 9 \) and \( Q = 10 \), which may be enough to capture local information. We also try large \( Q \), but when we set \( Q = 15 \), we find there’s only little improvement, and thus we set \( Q = 10 \) by default.

Effect of unidirectional cross attention. As discussed in Sec. 3.2, we find unidirectional cross attention makes ADCLR more effective to downstream tasks (with only the raw patches and [CLS] tokens as input). We conduct experiments by using unidirectional and bidirectional cross attention mechanisms and the results are given in Fig. 2(c). When \( Q = 1 \), ADCLR (bidir) drops about 3%~4% accuracy. When increasing the number of query view \( Q \), the accuracy of the downstream classification task drops notably. When we set \( Q = 196 \) (same with query token), we find ADCLR (bidir) only gets about 50% accuracy, while ADCLR (unidir) gets 68.8% accuracy.

Effect of query crops ratio \( s \). Since ADCLR introduces the query crops, we further study the performance with different local and global crop ratios. Following iBOT (Zhou et al., 2022), we conduct the experiments by tweaking \( s \), where \( s \) is the scale dividing the local and global crops. The query crops and local crops are sampled from (0.05, \( s \)). The global crops are sampled from (0.4, 1). We fix the other hyper-parameter and pretrain ADCLR with 100 epochs. We report the linear classification accuracy in Table 6. We evaluate crop ratios smaller than 0.2 because of the low resolution of query crop With a large ratio, low-resolution crops may lose lots of local information.
Table 7: Ablation of the number of local views.

<table>
<thead>
<tr>
<th>crop strategy</th>
<th>100 epochs</th>
<th></th>
<th></th>
<th>300 epochs</th>
<th></th>
<th></th>
<th>#mem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1 time</td>
<td>Top-1 time</td>
<td>#mem.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 × 224²</td>
<td>68.9</td>
<td>16.1h</td>
<td>73.9</td>
<td>47.6h</td>
<td>9.8G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 × 224² + 2 × 96²</td>
<td>72.2</td>
<td>17.7h</td>
<td>75.3</td>
<td>51.0h</td>
<td>11.1G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 × 224² + 6 × 96²</td>
<td>74.6</td>
<td>20.8h</td>
<td>76.7</td>
<td>60.9h</td>
<td>13.4G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 × 224² + 10 × 96²</td>
<td>75.1</td>
<td>24.7h</td>
<td>77.0</td>
<td>72.6h</td>
<td>16.7G</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Effect of local crops. We further conduct ablation on two 8-GPU machines on the number of local views as reported in Table 7. We fix the batch size 1024 and evaluate it pretrained with 100 and 300 epochs. The multi-crop strategy follows SwAV (Caron et al., 2020), i.e., 2 global views (224*224) and 2, 6, 10 local views (96*96). Under the two-view setting, ADCLR achieves 68.9% top-1 accuracy with 100 epochs pretraining. With the same setting, we reproduce the result of DINO, resulting 67.7% top-1 accuracy with 100 epochs pretraining, where our ADCLR outperforms DINO +1.2% accuracy. For multi-crops setting, we run DINO with 300 epochs, using $2 \times 224^2 + 10 \times 96^2$ crop strategy, resulting 76.2% (0.1% higher than reported in DINO) top-1 accuracy, ADCLR outperforms DINO by +0.8% top-1 accuracy. We find ADCLR gets higher accuracy under the two-crops strategy, which we guess using two global views for DINO is difficult to extract local information and the object in global views cannot be matched exactly. For ADCLR, the query tokens are cropped from the raw images with 16*16 pixels, which help capture accurate and dense information (see Fig. 1).

5 Further Discussion to Related Works

We discuss the relation of ADCLR against similar approaches to better position our work.

Relation to DINO (Caron et al., 2021). Connection. ADCLR adopts the learning framework of DINO (EMA update, sharpening and centering). The global branch of ADCLR is the same as DINO. Differences. On the basis of DINO, ADCLR proposes a local branch to learn spatial-sensitive information. Besides, technically, ADCLR proposes the unidirectional cross attention mechanism.

Relation to iBOT (Zhou et al., 2022). Connection. Both iBOT and ADCLR learn global and local information. Differences. iBOT learns the local information by adding MIM objective, which is now widely developed in MAE (He et al., 2021), SimMIM (Xie et al., 2021d). ADCLR learns the local information by maximizing the consistency of the same query crops in latent space (the variance is caused by the attention mechanism of two views). Compared with iBOT (masks patches), query tokens in ADCLR may be out of the views, which increases the difficulty of local contrasting.

Relation to SelfPatch (Yun et al., 2022). Connection. Both ADCLR and SelfPatch adopt DINO’s framework on the patch level to learn dense information. Differences. The idea of SelfPatch is very straightforward, i.e., regard top-k neighbors of query patch as positives, which may bring the noise (the selection of k). Besides, the local branch of SelfPatch requires input views has the same location, which decreases the learning difficulty. In contrast, ADCLR is simple yet also carefully designed. The local branch of ADCLR reserves the variance caused by augmentation (ADCLR can solve the positional shift caused by random resize and crop). Technically, SelfPatch designs the matching algorithm. For each query crop, it calculates the similarities and sorts them, which may bring a lot more complexity. ADCLR only introduces Q query patches, where $1 \leq Q \leq 10$. The complexity brought by ADCLR can be ignored (ViT-B/16 and ViT-S/16 divide the image into 196 raw patches.)

6 Conclusion

This paper proposes ADCLR, a new contrastive learning method aiming to capture accurate and local information for vision representation. By introducing “query crop”, ADCLR could easily match spatial location without complex operation, e.g., bi-linear interpolation. To make ADCLR more adaptable to downstream tasks and prevent collapse, we design unidirectional cross attention. Experiments on classification, detection, and segmentation show the effectiveness.

Limitations. Currently, the paradigm of “query crops” only supports low resolution (mapping the cropped region into small patches), which may lose semantic information. Hence, we think the multi-scale paradigm could help ADCLR a lot, which may be a good extension in future works.

References


Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.


Mr D Murahari Reddy, Mr Sk Masthan Basha, Mr M Chinnaiahgari Hari, and Mr N Penchalaiah. Dall-e: Creating images from text. UGC Care Group I Journal, 2021.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The naturalist species classification and detection dataset. In CVPR, 2018.


Published as a conference paper at ICLR 2023


A Proof for Theorem 1

Proof For simplicity, we consider in one layer single-head self-attention block, and remove the g, b re-scale parameters in layer normalization (Ba et al., 2016). Recall the self-attention mechanism in Transformer, i.e., $\text{Attn}(x, W_Q, W_K, W_V) = \text{Softmax}(\frac{(xW_Q)(xW_K)\top}{\sqrt{d}})xW_V$, where $W_Q$, $W_K$ and $W_V$ are learnable weights, and $\sqrt{d}$ is the hidden dimension. Note that $x \in \mathbb{R}^{(1+N+Q) \times d}$ includes $[CLS]$ token, raw patch sequence and query patch sequence. Let $s^{i,j} = \frac{(xW_Q)(xW_K)\top}{\sqrt{d}}$, where $s^{i,j} < \frac{e^{-1}}{(N+Q)e^{-1}}$, since the normalized $\|xW_Q\|_2 = 1$. Then, the input of two branches in contrastive learning becomes $[x_A^0, x_A^1, x_A^2, \ldots, x_A^Q, \ldots, x_A^{N+Q}]$ for view $A$ and $[x_B^0, x_B^1, x_B^2, \ldots, x_B^Q, \ldots, x_B^{N+Q}]$ for view $B$. Note that we have $x_A^i = x_B^i$ for $1 \leq i \leq Q$. Then, the embedding of i-th patch $z_A^i$ and $z_B^i$ through self-attention block can be written as:

$$z_A^i = (\sum_{i=0}^{N+Q} s^{i,j}_A \cdot x_A^j)W_V, \quad z_B^i = (\sum_{i=0}^{N+Q} s^{i,j}_B \cdot x_B^j)W_V, \quad 0 \leq i \leq N + Q \quad (7)$$

Recall the objective of negative-free contrastive learning, i.e., aligning the representations $\max \text{sim}(z_A^i, z_B^i)$ for $0 \leq i \leq Q$, where $z_0$ is the embedding of $[CLS]$ token. For simplicity, we use the mean square error (MSE) objective for analysis. Take the gradient into Eq. 7, we get:

$$\frac{\partial \mathcal{L}(z_A^i, z_B^i)}{\partial s^{i,j}_A} = \frac{\partial \mathcal{L}(z_A^i, z_B^i)}{\mathcal{H}(z_A^i, z_B^i)} \frac{\partial \mathcal{H}(z_A^i, z_B^i)}{\partial s^{i,j}_A} = \frac{\partial \|\sum_{j=0}^{N+Q} s^{i,j}_A x_A^j - \sum_{j=0}^{N+Q} s^{i,j}_B x_B^j\|_2^2}{\partial s^{i,j}_A} \quad (8)$$

where $s^{i,j}_A$ means attention value of i-th patch to j-th patch in view $A$ branch. Since $x$ is standardized and $Q >> N$, Then, the numerator of Eq. 8 can be written as $\|\sum_{j=0}^{Q} s^{i,j}_A x_A^j - \sum_{j=0}^{Q} s^{i,j}_B x_B^j\|_2^2$. Then, the network could easily get collapse solution, i.e., making $s^{i,j}_A$ for $0 \leq i, j \leq Q$, leading the $\mathcal{L}(z_A^i, z_B^i) = 0$, since $x_A^i = x_B^i$ for $0 \leq i \leq Q$. Then, we complete the proof.

B Implementations

Backbone. We use the standard ViT architecture (Dosovitskiy et al., 2020). It has several blocks composed of multi-head self-attention (MHSA) (Vaswani et al., 2017), MLP, and LayerNorm (Ba et al., 2016). In line with DINO and iBOT, we mainly adopt ViT-S/16 and ViT-B/16 as backbones.

Pretraining hyper-parameters. In line with DINO, we train with Adamw (Loshchilov & Hutter, 2018) and a batch size of 1024, distributed over 32 GPUs using ViT-S/16. The learning rate is linearly ramped up during the first 10 epochs to its base value determined with the following linear scaling rule (Chen et al., 2020a): $lr = 0.0005$, batchsize$=256$. After warmup, we decay the learning rate with a cosine schedule (Loshchilov & Hutter, 2016). The weight decay also follows a cosine scheduled from 0.04 to 0.4. The temperature $\tau$ is set to 0.1 while we use a linear warm-up for $\tau_i$ from 0.04 to 0.07 during the first 30 epochs. We follow the data augmentations of BYOL (Grill et al., 2020) (color jittering, Gaussian blur, and solarization) and multi-crop (Caron et al., 2020) with a bicubic interpolation to adapt the position embeddings to the scales. We set $\lambda = 0.5$ and query crop ratio from 0.05 to 0.15. The code with configuration details for reproducing will be publicly available.

Pytorch-liked code of unidirectional cross attention.

class Attention(nn.Module):
    def forward(self, cls, x, query):
        """
        :param cls: B, 1, C
        :param x: B, N, C
        :param query: B, Q, C
        """
        qkv = self.qkv(torch.cat([cls, x, query], dim=1)).reshape(B, 1+N+Q, 3, self.num_heads, C // self.num_heads).permute(2, 0, 3, 1, 4)
        q, k, v = qkv[0], qkv[1], qkv[2]
# cls and x forward
attn = (q[:, :, :1+N] @ k[:, :, :1+N].transpose(-2, -1)) *
        self.scale
attn = attn.softmax(dim=-1)
attn = self.attn_drop(attn)
x = (attn @ v[:, :, :1+N]).transpose(1, 2).reshape(B, N+1, C)

# query forward
attn = (q[:, :, N+1:] @ k[:, :, 1:N+1].transpose(-2, -1)) *
        self.scale
attn = attn.softmax(dim=-1)
attn = self.attn_drop(attn)
query = (attn @ v[:, :, 1:N+1]).transpose(1, 2).reshape(B, 1, C)
x = torch.cat([x, query], dim=1)
x = self.proj(x)
x = self.proj_drop(x)
return x[:, :1], x[:, 1:N+1], x[:, N+1:], attn