Self-supervised Learning of Contextualized Local Visual Embeddings

Anonymous ICCV submission

Paper ID ****

Abstract

We present Contextualized Local Visual Embeddings (CLoVE), a self-supervised convolutional-based method that learns representations suited for dense prediction tasks. CLoVE deviates from current methods and optimizes a single loss function that operates at the level of contextualized local embeddings learned from output feature maps of convolutional neural network (CNN) encoders. To learn contextualized embeddings, CLoVE proposes a normalized mult-head self-attention layer that combines local features from different parts of an image based on similarity. We extensively benchmark CLoVE's pre-trained representations on multiple datasets. CLoVE reaches state-of-the-art performance for CNN-based architectures in 4 dense prediction downstream tasks, including object detection, instance segmentation, keypoint detection, and dense pose estimation.

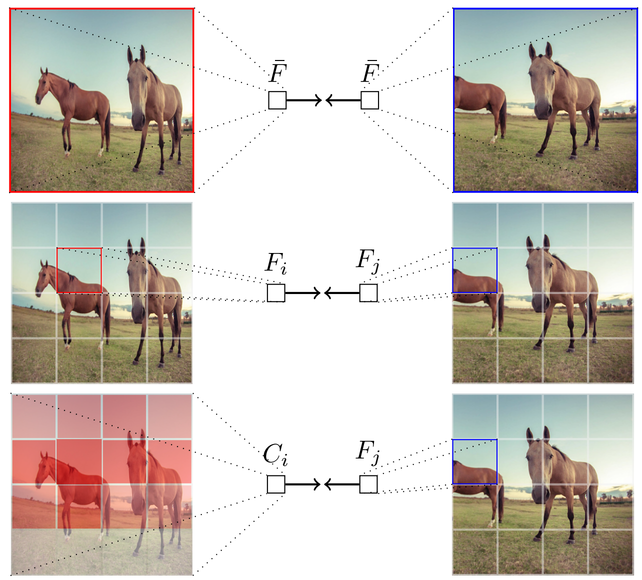

Figure 1. SSL strategies to learn representations. Embedding similarity optimization over global representations (top), local representations (middle), and contextualized embeddings (bottom).

1. Introduction

Self-supervised learning (SSL) has become an essential technique for learning downstream tasks. For tasks in which data annotation is pricey or even impossible to acquire, a round of self-supervised pre-training prior to learning the downstream task of interest can significantly enhance the system's final performance and reduce costs with data annotation.

In computer vision, one of the main advantages of SSL [10, 16, 17, 22] over generative models [15, 23, 28], is the avoidance of reconstructing the input signal. Typically, generative models optimize a cost function in the pixel space seeking to reconstruct the input signal with high fidelity. Besides the high computing costs of operating in the pixel space, these methods assume that every pixel in the input image matters equally. However, from the representation learning perspective, this property may not be necessary.

Instead, the SSL approach of working at the embedding level allows SSL methods to learn representations that discard useless information. This strategy can be precious for learning downstream tasks since much of the details of an image may be useless for solving many downstream tasks. For instance, if the task of interest only requires a global signal, such as the class information, given a fixed-size feature vector, the encoder may be encouraged to discard low-level details, such as position, background, and orientation, in favor of features associated with the class information.

Classic convolutional neural networks (CNNs) were primarily designed to address classification tasks. CNNs decimate the spatial dimensions of the input in favor of learning dense feature maps that are collapsed to a single global representation vector before going to a classifier layer. This engineering tendency encourages the convolutional encoder to discard fine-grained information from the input. In fact, that is why many segmentation models [9, 21] attempt to reconstruct the input image, which can be viewed as learning the low-level details lost in the encoding process.

We argue that current SSL methods, based on CNN backbones, inherit the same architecture designs and suffer from similar problems. Collapsing the output feature maps of a CNN encoder into a global-level representation using an aggregation function, such as the average, encourages the encoder to discard low-level details that are crucial for solving dense prediction tasks, such as detection and segmentation.

Based on these assumptions, we conjecture that CNN-based SSL methods carry an engineering bias toward downstream tasks that do not require low-level information from the input. Such biases are also enforced by evaluation protocols that primarily assess the classification power of the learned representation. For these reasons, state-of-the-art SSL methods perform much better in classification tasks than downstream tasks requiring dense predictions.

To close this gap, we propose an algorithmic approach that focuses on learning contextualized visual embeddings. Contextualized embeddings combine local features of an image based on self-similarities. Instead of aggregating local feature maps into a global vector using an arithmetic average that attributes equal weights to each local feature, we bootstrap multiple prediction vectors (one for each local feature) based on learned weighted averages that capture contextualized information from similar regions of the input image, as illustrated in Figure 1. This way, we can bootstrap prediction vectors that aggregate multiple areas of an image view that share semantic meaning to predict local parts of a different view of the same image. Our method, **C**ontextualized **Lo**cal **V**isual **E**mbeddings (CLoVE), is designed to learn representations that preserve local information from the input by finding correlations among similar regions of a view to predict local parts of a different view. The motivation is to learn representations that excel at solving downstream dense prediction tasks.

Traditional SSL methods primarily focus on optimizing global representations of different views on an image [3, 16, 17]. When training CNN backbones, the output local feature map is collapsed using an average function and treated as a global image representation. Conversely, current SSL methods designed for dense prediction representation learning [24, 31, 37] either optimize for local representations or combine local and global objectives. In contrast, CLoVE does not optimize directly for local or global representations. Instead, it poses the representation learning problems at the level of contextualized local embeddings. In essence, we propose an objective function that predicts a target representation from a local part of a view using a combination of correlated local embeddings from another view. Figure 2 illustrates our architecture.

Our contributions are twofold. Firstly, we introduce a novel method that does not optimize for local or global embeddings. Secondly, we propose a variation of the self-attention algorithm and integrate it into CNN architectures. This way, we enable the learning of representations that effectively retain local information from the input and capture long-range dependencies from representations that share semantic meaning. This integration empowers our approach to excel in dense prediction downstream tasks, where fine-grained details play a vital role in achieving high performance and accuracy. Our method is extensively evaluated and proves its effectiveness in downstream tasks, including object and keypoint detection, segmentation, and pose estimation.

2. Related work

Recent self-supervised representation learning methods follow a similar framework composed of the following building blocks: (1) a joint-embedding architecture, (2) a pretext task, and (3) a similarity-based loss function. The joint-embedding architecture may be pure siamese [5] or follow a teacher-student [11] architecture with a separate momentum encoder that usually does not receive gradients. Among many proposed pretext tasks, one that stands out is instance discrimination [1, 36]. In instance discrimination, we task a deep neural network to find a pair of representations from different views of the same image among a set of negative pairs where the representation from the anchor image is paired with representations from random images. Lastly, the similarity loss function may be contrastive [10, 17, 27], in which InfoNCE [22] is a popular choice, or non-contrastive [12, 16].

SSL methods differ in how they optimize the embedding space. While a group of methods directly optimize the representations using a similarity loss function [10, 17, 42], others discretize the embedding space by learning prototypes [2, 6, 7, 26]. Despite differences, these methods are designed to learn global representations from the input image. When the feature extractor is represented as a CNN, the feature map from the last convolutional layer is collapsed into a single vector through a global average pooling operation. If a Transformer backbone is used, the class-token representation is optimized as a global feature vector [8, 13]. These methods generally learn powerful, invariant representations for classification problems but do not perform as well when the downstream task requires localization and/or low-level details.

Recently, we have witnessed the emergence of methods designed for dense prediction tasks [4, 24, 31, 39]. Generally, these methods take one of two approaches to learn representations (1) they pose the learning problem at the level of local embeddings [24], or (2) they optimize for global and local embeddings jointly [4, 31, 37, 39]. Most methods fall into the second category, where two loss functions are minimized, one that operates on representations from the full view and another on representations from local parts of the image. The two loss functions are linearly combined to a final objective and jointly optimized. Some evidence suggests a trade-off between global and local feature learning for SSL [4, 31], which might explain the popular algorithmic design. We can view this approach as an extension of current SSL methods, allowing them to trade off global and local characteristics in their learning features.

Among methods that pose the learning problem at the local feature level, the approach developed by Pinheiro et al. [24] stands out. The method learns dense (pixel-level) representations by exploring contrastive learning over local features that map to the same pixel across different views in the input space. The architecture learns local feature vectors by reconstructing the feature map using a decoder model and applies contrastive learning at a higher level of feature reconstruction.

Among methods that combine global and local objectives, recent work [31, 32, 37] used the InfoNCE loss to learn global and local representations and can be viewed as extensions of MoCo [17]. Wang et al. [31] proposed a loss function that performs contrastive learning at the level of local features. To match local features across different views, they use a cosine similarity function where a local feature from one view takes the most similar local feature from the other view as its target. Similarly, Xiao et al. [37] proposed a region-level contrastive loss that relies on intersected regions between the two views of an image. Over intermediate layers of a convolutional encoder, the overlapping areas (feature maps) are processed by a fixed-sized window and fed to a Precise RoI Pooling [20] layer creating a feature vector from the region. In both cases, the local loss is implemented using the InfoNCE loss and jointly optimized with the global MoCo-style objective.

Xie et al. [39] proposed a non-contrastive local objective that can be viewed as an augmentation to the BYOL [16] global loss. They proposed the Pixel-to-Propagation module. A form of attention layer that creates contextualized local embeddings by combining local features in a vicinity. Lastly, Bardes et al. [4] extended the VicReg [3] method and applied the Variance-invariance-covariance regularization (VICReg) loss to learn global and local features.

Contrast to previous approaches. Our method differs from contemporary work in essential aspects. One of the main differences between CLoVE and existing approaches is the departure of jointly optimizing global and local objectives, thus avoiding the global/local feature learning trade-off. Instead, we learn multi-head self-attention layers that can bootstrap contextualized local embeddings that serve as predictions to target local features.

CLoVE may be regarded as similar to PixPro [39]. However, there are important differences between the two approaches. CLoVE combines multi-head self-attention layers, usually employed in transformers, to convolutional architectures in a contextualized local feature learning framework. On the other hand, the Pixel-to-Propagation module [39] differs from CLoVE in important aspects. Namely, (1) it does not learn multiple heads, (2) it does not learn transformation matrices for query, key, and value tensors, and (3) it does not normalize the result attention scores. Moreover, Xie et al. [39] combined a loss function at the local embedding with the standard BYOL global objective in a non-contrastive manner. Conversely, CLoVE does not work directly with global or local objectives and employs a ranking margin loss.

Unlike previous work [24], our architecture works directly at the feature map level and does not attempt to reconstruct local features. In contrast to the approach developed by Wang et al. [31], our strategy avoids the noisy process of choosing the most similar local embedding as the target. Instead, we match representations from which their center pixels lie within a vicinity in the pixel space.

3. Learning contextualized local representations

We strive to learn visual features that retain fine-grained details from the input and therefore are suited for dense prediction tasks. Unlike other methods, CLoVE does not optimize a global or a local loss function (or their combination). Instead, the learning problem is posed at the contextualized embeddings level, learned from feature maps of CNN encoders. In this framework, we use local features as target representations, and to predict such targets, we learn vectors that combine local features in a vicinity based on learned self-similarities. In essence, contextualized embeddings are a mixture of local, semantically similar features from different parts of a view. Local features are combined into a single prediction based on their similarity to the anchor local feature. Intuitively, this strategy allows learning richer prediction vectors that encode many similar parts of an image view to predict a localized portion of another view.

3.1. Preliminares

Given an image $x \in \mathbb{R}^{3\times H\times W}$ with no supervision, we create views of x, denoted $x^1 = \mathcal{T}(x)$ and $x^2 = \mathcal{T}(x)$, where $\mathcal{T}(\cdot)$ is a stochastic function that applies a set of random geometric and intensity transformations to x. Such transformations include random flips, color distortions, and cropping. In practice, we can work with many views, but for simplicity, we constrain the number of views to $N_v = 2$.

Each view is independently forwarded through a student encoder f_s, and a teacher encoder f_t. The encoders are composed of a feature extractor, *e.g.*, a CNN encoder, and a projection head represented as a multi-layer perceptron (MLP). Following previous work [16, 17], the teacher encoder f_t does not receive gradient updates. Instead, the weights θ_t are updated using a moving average of the weights θ_s, such as $\theta_t = \alpha\theta_t + (1-\alpha)\theta_s$, where α is the weight.

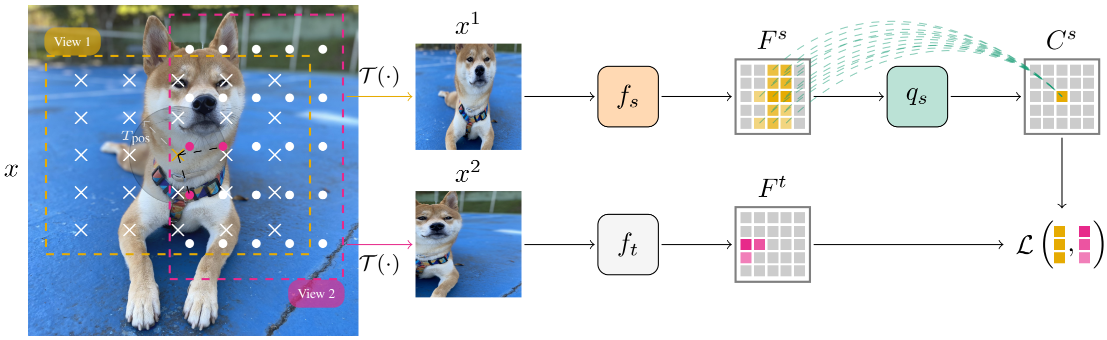

Figure 2. Views x^1 and x^2 are fed to student and teacher encoders f_s and f_t, to extract local feature maps F^s and F^t, respectively. The predictor q_s takes the local features F^s and outputs contextualized embeddings C^s by combining local features based on self-similarities. We define a grid of points proportional to the output feature map in each view. Points in one view are paired with points in the other based on distance in the ambient space. Selected points are mapped to the feature space and used to match embeddings in C^s with targets in F^s.

For each view, we obtain a tensor of projected local feature maps $F = f(x^v)$, for $v \in [0, 1]$. These local features correspond to the output feature map of an intermediate layer of the CNN feature extractor, projected to a lower dimensional space, and have a general shape of $F \in \mathbb{R}^{N \times D \times F_h \times F_w}$, where N is the batch size, D is the feature dimensionality, and F_h and F_w are the spatial dimensions of the feature map.

We can view the projected local features in F as a sequence of embeddings, $F \in \mathbb{R}^{N \times D \times L}$, where L is the sequence length $L = F_h \times F_w$. Traditional SSL methods take the feature maps from the CNN feature extractor (prior to projection) and collapse them using a global average operation to obtain a global representation. The global feature is fed to a projection head and then to a similarity-based loss function, as illustrated in Figure 1 (top). On the other hand, local SSL methods either maximize agreement between local embeddings or combine local and global objectives [4, 31, 39]. In a different direction, CLoVE learns contextualized representations through self-attention layers operating on local embeddings of a view.

Next, we detail how we extract dense self-supervision from image views and our contextualized loss function.

3.2. Pixel-to-representation neighborhood matching

To learn representations that retain low-level features, we need targets that contain such properties. In other words, we must bootstrap dense self-supervised signals to use as targets in our loss function. One way is to track pixels' locations as we create views x^1 and x^2. If two views share an intersected area, the pixels in this region represent the same part in the original image. However, scaling and resizing may push these pixels to random locations during the view's creation. Instead of matching exact pixels across views, we can look for pixels' neighbors. This strategy explores the pixel spatial locality inductive bias in which nearby pixels represent similar contexts, hence should have similar representations. Once we match pixels across views based on neighborhood distances, we can map the pixels' locations to the feature space to index local features in the loss function.

We define I^1 and I^2 as lists of 2D points in the pixel space. Points in I^1 are defined over the first view, and points in I^2 over the second. For each point I_i^1 in the first view, we look for pixel correspondences in the second view by extracting nearby points in I^2 that lie within a similarity region. Accordingly, we define M as the set of all pairs (I_i^1, I_j^2) such that the euclidian distance between points I_i^1 and I_j^2 is smaller than a threshold T_{pos}, such as

$$M = \left\{ \left(I_i^1, I_j^2\right) \mid d\left(I_i^1, I_j^2\right) < T_{\text{pos}} \right\}, \tag{1}$$

where $d(a, b) = \sqrt{\sum_{i=0}^{2} (a_i, b_i)^2}$.

Next, we map the points in M from the pixel space to the feature space. Each point in M is mapped to its respective local embedding in the feature map of the CNN encoder. Therefore, the pair of points in M now represent a pair of indices matching features from view 1 to view 2. This process is depicted in Figure 2.

The Pixel-to-Neighborhood matching strategy will pair at most $p = |F|$ points for each local embedding, where F represents the projected feature map from the CNN encoder. For a ResNet-50 encoder, we define 49 points in a grid structure that are mapped to each of the 7×7 local features in F, as described in Section 6.

One advantage of this matching algorithm is that we do not need to force views to share an intersected region. Local representations from different views that do not intersect can still be paired if they are close enough in the pixel space. Moreover, the choice of T_{pos} matters since it controls the average number of target local representations. Intuitively, if T_{pos} is too high, a pixel I_i^1 might consider all pixels in I^2 as neighbors. As a result, it invalidates the spatial locality inductive bias present in natural images. On the other hand, if T_{pos} is too low, it limits the target space as the spatial locality bias is not explored to its fullest, as described in Section 5.3.

3.3. Predicting local embeddings with contextualized vectors

At this point, we can match local features across different views on an image using the feature indices in M. However, this learning objective would fail to learn long-range dependencies. Intuitively, if an object occupies a large portion of an image, we want to maximize the agreement between all semantically meaningful parts of the object or region and its local target embedding. To accomplish this strategy, each local feature of the first view can interact with its neighboring local features to learn similarity patterns. This way, local features exhibiting strong similarity are combined into a single contextualized vector and used to predict the local target embedding from another view.

To learn contextualized embeddings, we propose a predictor head q_s that receives the output feature map F^s from the student and apply a Normalized Multi-Head Self-Attention (NMHSA) layer to obtain $C^s = q_s(F^s)$, where $q_s(F^s) = \text{NMHSA}(F^s)$. We use the matching feature indices in M to select contextualized predictions and target local features from C^s and F^t, respectively. Then, we maximize agreement between contextualized and local embeddings by minimizing the margin ranking loss defined as,

$$\mathcal{L} = \sum_{(i,j)\in M} \max\left(0, -\lambda\sigma\left(C_i^s, F_j^t\right) + \sigma\left(C_i^s, F_{\text{neg}}^t\right) + \mu\right), \tag{2}$$

where μ is the margin, $\sigma(a,b) = \frac{xy}{\|x\|_2\|y\|_2}$ is the cosine similarity function and $\|\cdot\|_2$ is the ℓ_2 norm.

For each pair of matching features indexed by (I_i^1, I_j^2), we maximize agreement between contextualized representations from one view and local embeddings from the other.

To bootstrap the negative representation F_{neg}, we follow a similar strategy proposed by Wang et al. [30]. We compute the cosine similarity between the contextualized predictions C^s and all local representations from the opposing view F^t. Then, we select the top-k most offending local representations (higher similarities scores) from F^t, discard the most similar one, and take the average of the resulting vectors. Intuitively, we discard the most offending local feature from F^t because it could represent a false negative. This selection strategy can be viewed as finding a *negative* region within the image that is not correlated with the contextualized predictor. The size of the negative region is controlled by k and set as $k = 10$. We show in Section 5.4 that choosing negatives within the image is most beneficial to the learned representation as selecting negatives across different images.

3.4. The normalized attention head

We can view the self-attention mechanism as combining similar local areas of a view. Intuitively, to successfully predict the local region of the second view, the self-attention must combine the local features of the first view in a way that similar content has a strong contribution and dissimilar content has a weak contribution to the contextualized embedding.

In practice, we learn 8 self-attention heads, where head$[i]$ = Attention(F^sW^q, F^sW^k, F^sW^v) and Attention(Q,K,V) = softmax $\left(\frac{\sigma(Q,K^T)}{\tau}\right)V$. We show in Section 5.2 that, in practice, normalizing queries and keys before computing the attention scores improves the final downstream tasks' performance.

From an intuitive perspective, by matching contextualized representations with local embeddings (based on pixel spatial locality), the network learns to (1) attend to similar regions in the input and (2) disregard local embeddings representing different contexts in the same view. This process optimizes multiple prediction subtasks, *i.e.*, for each local feature F_i^s, there is a contextualized representation C_i^s. As a result, the learned representations retain fine-grain details from the input.

4. Main experiments

To assess how well CLoVE's pre-trained representations transfer to dense prediction tasks, we fine-tuned detection and segmentation models, using `Detectron2` [35], on Pascal VOC07, COCO, LVIS, and Cityscapes datasets. For the competing methods, we used the officially released model checkpoints and reported performance metrics from their papers if the same evaluation protocol. Otherwise, we ran experiments in-house. We pre-trained CLoVE on the ImageNet-1M dataset for 200 and 400 epochs and compare its performance against state-of-the-art SSL methods on various downstream tasks such as object detection, instance segmentation, keypoint detection, and dense pose estimation. The experiments report average performance across 5 independent runs. We highlight the top-1 performing methods in **bold** and top-2 underlined.

COCO detection and instance segmentation. Tables 1 and 4 compare CLoVE's performance using the R50-C4 and R50-FPN backbones against other methods. For the two backbones, CLoVE achieved top-1 performance across both tasks. Additionally, CLoVE reached top-2 performance in 5 out of the 6 for R50-C4 and 4 out of 6 for R50-FPN in low-resource training settings.

Cityscapes instance segmentation. In Table 3, CLoVE achieves an average improvement of **+1.4** AP over Pix-

Table 1. Obj. detection and segmentation on COCO (R50-C4).

Method	ep	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mb}	AP^{mb}_{50}	AP^{mb}_{75}
Supervised	100	38.2	58.2	41.2	33.3	54.7	35.2
Rand init	–	26.4	44.0	27.8	29.3	46.9	30.8
ReSim [37]	200	39.7	59.0	43.0	34.6	55.9	37.1
InsCon [40]	200	40.3	60.0	43.5	35.1	56.7	37.6
PixPro [39]	400	40.5	59.8	44.0	35.4	56.9	37.7
DetCo [38]	200	39.8	59.7	43.0	34.7	56.3	36.7
SlotCon [33]	200	39.9	59.8	43.0	34.9	56.5	37.3
CLoVE	200	40.6	60.0	44.1	35.4	56.8	37.8
	400	**41.0**	**60.3**	**44.2**	**35.5**	**57.2**	**38.1**

Table 2. Obj. detection and segmentation on COCO (R50-FPN).

Method	ep	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mb}	AP^{mb}_{50}	AP^{mb}_{75}
Supervised	100	38.9	59.6	42.7	35.4	56.5	38.1
Rand init	–	32.8	51.0	35.3	28.5	46.8	30.4
DenseCL [31]	200	39.4	59.9	42.7	35.6	56.7	38.2
ReSim [37]	200	39.3	59.7	43.1	35.7	56.7	38.1
PixPro [39]	400	39.8	59.5	43.7	36.1	56.5	38.9
SetSim [32]	200	40.2	60.7	43.9	36.4	57.7	39.0
VICRegL [4]	300	37.3	57.6	40.7	34.1	54.7	36.5
CLoVE	200	40.8	60.5	45.0	36.8	57.6	39.8
	400	**41.2**	**61.1**	**45.0**	**37.1**	**58.1**	**40.1**

Table 3. Instance segmentation on Cityscapes (R50-FPN).

Method	ep	AP	AP_{50}
Supervised	100	26.5	52.9
Rand init	–	19.9	40.7
DenseCL [31]	200	33.1	61.7
PixPro [39]	400	35.8	63.7
VICRegL [4]	300	29.8	58.5
SlotCon [34]	200	35.2	63.8
CLoVE	200	35.7	64.1
	400	**37.2**	**65.3**

Pro [39], and **+10.7** AP over the supervised baseline.

LVIS object detection and instance segmentation. LVIS is a dataset for long-tail object recognition. It contains more than 1200 classes and more than 2M high-quality instance segmentation masks. In Table 4, CLoVE 200 epoch model performs similarly to PixPro. The 400 epoch model beats competitors by a small margin and reaches a performance gap of **+4** points against the supervised baseline for all metrics.

COCO keypoint detection. In Table 5, CLoVE performs comparably to other SSL methods and surpasses the supervised baseline by **+1.7** average AP. For keypoint detection, we noticed that the CLoVE 400 epoch model did not improve over the 200 epoch model. In Figure 3, we report qualitative results for keypoint detection on randomly chosen images.

Table 4. Obj. detection and segmentation on LVIS (R50-FPN).

Method	ep	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mb}	AP^{mb}_{50}	AP^{mb}_{75}
Supervised	100	20.2	33.4	21.4	19.6	31.2	20.8
Rand init	–	12.4	21.8	12.5	12.1	20.2	12.5
DenseCL [31]	200	20.4	33.5	21.4	19.9	31.5	20.9
PixPro [39]	400	23.8	38.2	25.2	23.3	36.1	24.7
SlotCon [33]	200	23.2	37.6	24.3	22.9	35.6	24.3
VICRegL [4]	200	7.0	13.4	6.4	7.4	12.7	7.3
CLoVE	200	23.6	37.7	25.2	23.3	35.9	24.8
	400	**24.3**	**38.8**	**25.8**	**23.9**	**36.7**	**25.3**

Table 5. Keypoint detection on COCO (R50-FPN).

Method	ep	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
Supervised	100	65.3	87.0	71.3
Rand init	–	63.0	85.1	68.4
DenseCL [31]	200	66.3	87.1	71.9
PixPro [39]	400	66.6	87.2	73.0
ReSim [29]	200	66.3	87.2	72.4
SetSim [32]	200	66.7	**87.8**	72.4
SlotCon [34]	200	66.5	87.5	72.5
CLoVE	200	66.9	87.5	73.2
	400	**67.0**	87.4	**73.3**

Table 6. Object detection on Pascal VOC (R50-C4).

Method	ep	AP	AP_{50}	AP_{75}
Supervised	100	53.5	81.3	58.8
Rand init	–	33.8	60.2	33.1
DenseCL [31]	200	58.7	82.8	65.2
ReSim [29]	200	58.7	83.1	66.3
InsCon [40]	200	59.1	83.6	66.6
PixPro [39]	400	60.0	**83.8**	67.7
cp2 [29]	600	56.9	82.3	63.6
SlotCon [34]	200	57.3	82.9	64.3
SetSim [32]	200	59.1	83.2	66.1
CLoVE	200	**60.1**	83.7	67.7
	400	59.9	**83.8**	**67.8**

Pascal VOC Object Detection. In Table 6, CLoVE 200 epoch model performs comparably with PixPro [39]. Similarly to keypoint detection, the CLoVE 400 epoch model did not improve upon the 200 epoch version.

COCO dense pose estimation. In Table 7, CLoVE average performance beats supervised models trained on ResNet-50 and ResNet-100 backbones. Figure 3 shows CLoVE's qualitative results for the dense-pose estimation downstream task.

Notes on VICRegL. VICRegL performance was surprisingly below expectations in many downstream tasks. While Bardes et al. [4] reported AP of **59.5** for the same protocol and model (`resnet50_alpha0p75.pth`) we used, our experiments resulted in AP of **27.6** on VOC07. Additionally, there is an open issue on VICRegL's official

Table 7. Dense pose estimation on COCO (R50-FPN).

Method	ep	AP^{bb}	AP^{mb}	AP^{gps}	AP^{gpsm}
Supervised (R50) [35]	100	61.2	67.2	63.7	65.3
Supervised (R101) [35]	100	62.3	67.8	64.5	66.2
DenseCL [31]	200	63.0	67.7	65.7	66.7
PixPro [39]	400	63.1	**68.3**	66.2	67.4
SlotCon [34]	200	62.8	67.4	65.3	66.4
CLoVE	200	**63.2**	68.2	**66.6**	**67.5**
	400	**63.2**	**68.3**	66.3	67.3

Table 8. Contrastive *vs*. non-contrastive loss functions and the effect of multi-crop augmentation.

Loss	multi-crop	AP	AP_{50}	AP_{75}
ℓ_2	✗	58.6	82.8	**66.2**
	✓	58.3	82.9	65.3
Rank	✗	58.5	82.8	65.6
	✓	**58.8**	**83.3**	65.9

GitHub repo reporting the same reproducibility problem with similar results.

5. Ablations

To ablate the main hyperparameters of our model, we pre-trained CLoVE on the ImageNet-1M dataset for 50 epochs and reported average performance results (3 independent runs) on Pascal VOC07 object detection.

5.1. Multi-crop and the choice of loss function

In Table 8, we explore two loss functions that could be used in CLoVE's learning framework: the non-contrastive ℓ_2-norm dot product and the ranking margin loss (2). Moreover, we evaluate the effect of multi-crop augmentation on both loss functions. The ℓ_2-normalized dot product loss, proposed by Grill et al. [16] and used in PixPro [39], performs well with two views. However, performance decreases when multi-crop is employed. On the other hand, the ranking loss performs well in both setups as it can extract extra performance from multi-crop augmentation.

5.2. Normalized multi-head self-attention

We propose a variation of the MHSA layer employed in Vision Transformers [14]. Specifically, we normalize queries and keys before computing the attention scores. By normalizing the vector's magnitudes, we constrain the similarity scores to -1.0 and 1.0, which in practice, avoids training instabilities and improves downstream task performance, *cf*. Table 9.

5.3. Bootstrapping self-supervised signals

To match local representations across different views of an image, we explore the spatial locality inductive bias present in natural images and expand it to the feature space.

Table 9. Normalized multi-head self-attention (NMHSA) performs slightly better than regular MHSA.

Method	AP	AP_{50}	AP_{75}
MHSA	58.3	83.1	65.8
NMHSA	**58.7**	**83.3**	**65.9**

Table 10. Negative sampling strategies for contrastive learning.

Method	queue	AP	AP_{50}	AP_{75}
Inter	✓	57.4	82.5	63.6
Inter (avg)	✓	57.5	82.8	64.7
Intra	✗	**58.7**	**83.3**	**65.9**

Table 11. The effect of T_{pos} on the learned representations.

	0.5	0.6	0.7	0.8	0.9
T_{pos}	57.0	58.3	**58.5**	58.1	57.7

Intuitively, if two distinct pixels lie within a distance threshold T_{pos}, we assume their representations encode similar information. In Table 11, we explore the effect of the distance threshold used to identify pixels as neighbors across different views. As shown, too small or too large values for T_{pos} invalidate the inductive bias assumption and harms the learned representations. Figure 2 pictorially describes the T_{pos} threshold.

5.4. Exploring negative sampling strategies

In Table 10, we explore three negative sampling strategies for CLoVE's loss function (2). For two strategies, we utilize an extra queue containing 16 384 representations as a source of negatives. In the first strategy (inter), at each training iteration, we randomly take one local representation from the output feature map of the teacher branch and store it in the queue. Older representations in the queue are discarded in favor of new ones. This way, the queue holds local representations from multiple images. In the second strategy (inter avg), we aggregate the feature map into a single vector using a global average operator. Lastly, we use the local features without positive matchings from within the view as negatives. Since this strategy does not require negatives from other images (no queue), we call it intra-negative. As shown in Table 10, the intra-negative strategy outperforms the other ones in VOC07 and is CLoVE's default strategy.

6. Implementation details

We use the ResNet-50 [19] architecture without the last fully connected and global average pooling layers as the feature extractor. Following, the projection head is a two-layer MLP with 4096 hidden units, ReLU, batch normalization, and an output dimension of 256. To create views, we follow Grill et al.'s [16] protocol.

We forward an image view $x \in \mathbb{R}^{3\times224\times224}$ and ob-

Figure 3. Qualitative results for keypoint detection (top row) and dense pose estimation (bottom row).

tain a feature map $F \in \mathbb{R}^{256\times7\times7}$. The contextualized prediction head q_s implements the Normalized Multi-Head Self-Attention layer. It receives the feature map as input and trains 8 parallel attention heads. Each attention head learns independent query, key, and value matrices, $W^q, W^k, W^v \in \mathbb{R}^{256\times32}$. To compute the attention scores, we normalize the projected queries and keys to unit vectors. The output of each head is concatenated (in the feature dimension) and passed through a linear output layer whose output has the same shape as the input.

CLoVE is trained using 4 NVIDIA A100 GPUs, with a total batch size of 2048 images, using the LARS [41] optimizer, weight decay of 2×10^{-5} and learning rate of 1.0 with a cosine decay schedule. In practice, the margin value in (2) is set to $\mu = 100$.

6.1. Evaluation protocols

COCO detection and instance segmentation. We followed the protocol from He et al. [17] and fine-tuned all layers of a Mask-RCNN [18] on the `train2017` set (∼118k images) and evaluated on `val2017`, using the 1× schedule (∼12 epochs).

Cityscapes instance segmentation. We followed the `mask_rcnn_R_50_FPN.yaml` config file from `Detectron2` [35], without changes, and fine-tuned all layers of a Mask-RCNN (R50-FPN backbone) for 24k iterations, with a global batch size of 32 images (8 per GPU), and a learning rate of 0.01.

LVIS object detection and instance segmentation. We followed the `mask_rcnn_R_50_FPN_1x.yaml` config file for LVISv1 instance segmentation from `Detectron2`, with no BN, and fine-tuned a Mask R-CNN (R50-FPN) on `lvis_v1_train` for 180k iterations (1 × schedule) with a batch size of 16 (4 images per GPU), a learning rate of 0.001 and evaluated on `lvis_v1_val`.

COCO keypoint detection. We used the keypoint implementation of Mask R-CNN (R50-FPN) from `Detectron2`, fined tuned on `keypoints_coco_2017_train`, and evaluated on `keypoints_coco_2017_val` for 90k iterations (1 × schedule), a batch size of 16 (4 images per batch), a learning rate of 0.02, and with enabled BN.

Pascal VOC Object Detection. We followed He et al.'s [17] protocol and fine-tuned all layers of a Faster R-CNN [25] (R50-C4) on `trainval07+12` (∼16.5k images) for 24k iterations and evaluated on `test2007`.

COCO dense pose estimation. We followed the DensePose [35] project from `Detectron2` and fine-tuned a Faster R-CNN (R50-FPN) backbone using CLoVE's pre-trained representations (1 × schedule). Specifically, we used the `densepose_rcnn_R_50_FPN_s1x.yaml` config file from the `Detectron2` repository, with BN enabled.

7. Conclusions

We presented **C**ontextualized **Lo**cal **V**isual **E**mbeddings (CLoVE), a self-supervised method designed to learn representations to solve dense prediction tasks. CLoVE combines the multi-head self-attention layer commonly used in the Transformer model with convolutional backbones to learn prediction vectors that combine multiple similar areas of a view into a contextualized vector used to predict a local part of another view. We empirically validate our design choices through a detailed ablative study of CLoVE's main hyperparameters. Additionally, we extensively benchmarked CLoVE in many downstream dense prediction tasks such as object detection, instance segmentation, keypoint detection, and dense pose estimation. CLoVE pre-trained representations showed robust performance against state-of-the-art SSL methods and supervised baselines.

References

[1] Dosovitskiy Alexey, Philipp Fischer, Jost Tobias, Martin Riedmiller Springenberg, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. In *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 99, 2015. 2

[2] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *Inter. Conf. Learn. Represent. (ICLR)*, 2019. 2

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *Inter. Conf. Learn. Represent. (ICLR)*, 2021. 2, 3

[4] Adrien Bardes, Jean Ponce, and Yann LeCun. VICRegL: Self-supervised learning of local visual features. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, 2022. 2, 3, 4, 6

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, volume 6, 1993. 2

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conf. Comput. Vis. (ECCV)*, pages 132–149, 2018. 2

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, volume 33, pages 9912–9924, 2020. 2

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE/CVF Inter. Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 9650–9660, 2021. 2

[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conf. Comput. Vis. (ECCV)*, pages 801–818, 2018. 1

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Inter. Conf. Mach. Learn. (ICML)*, pages 1597–1607. PMLR, 2020. 1, 2

[11] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, volume 33, pages 22243–22255, 2020. 2

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, pages 15750–15758, 2021. 2

[13] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Inter. Conf. Learn. Represent. (ICLR)*, pages 9640–9649, 2021. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Inter. Conf. Learn. Represent. (ICLR)*, 2021. 7

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020. 1

[16] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, volume 33, pages 21271–21284, 2020. 1, 2, 3, 7

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, pages 9729–9738, 2020. 1, 2, 3, 8

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE Inter. Conf. Comput. Vis. (ICCV)*, pages 2961–2969, 2017. 8

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Inter. Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 770–778, 2016. 7

[20] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *European Conf. Comput. Vis. (ECCV)*, pages 784–799, 2018. 3

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Inter. Conf. Comput. Vis. (ICCV)*, pages 3431–3440, 2015. 1

[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1, 2

[23] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE/CVF Inter. Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 2536–2544, 2016. 1

[24] Pedro O.O. Pinheiro, Amjad Almahairi, Ryan Benmalek, Florian Golemo, and Aaron C Courville. Unsupervised learning of dense visual representations. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, volume 33, pages 4489–4500, 2020. 2, 3

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Adv. Neural Inf. Process. Sys. (NeurIPS)*, volume 28. Curran Associates, Inc., 2015. 8

[26] Thalles Silva and Adín Ramírez Rivera. Representation learning via consistent assignment of views to clusters. In *ACM/SIGAPP Symp. Appl. Comp. (SAC)*, pages 987–994, 2022. 2

[27] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Adv. Neural Inf. Process. Sys. (NeurIPS)*, volume 33, pages 6827–

6839. Curran Associates, Inc., 2020. 2

[28] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Inter. Conf. Mach. Learn. (ICML)*, pages 1096–1103, 2008. 1

[29] Feng Wang, Huiyu Wang, Chen Wei, Alan Yuille, and Wei Shen. Cp 2: Copy-paste contrastive pretraining for semantic segmentation. In *European Conf. Comput. Vis. (ECCV)*, pages 499–515. Springer, 2022. 6

[30] Guangrun Wang, Keze Wang, Guangcong Wang, Philip HS Torr, and Liang Lin. Solving inefficiency of self-supervised representation learning. In *IEEE Inter. Conf. Comput. Vis. (ICCV)*, pages 9505–9515, 2021. 5

[31] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, pages 3024–3033, 2021. 2, 3, 4, 6, 7

[32] Zhaoqing Wang, Qiang Li, Guoxin Zhang, Pengfei Wan, Wen Zheng, Nannan Wang, Mingming Gong, and Tongliang Liu. Exploring set similarity for dense self-supervised representation learning. In *IEEE/CVF Inter. Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 16590–16599, 2022. 3, 6

[33] Xin Wen, Bingchen Zhao, Anlin Zheng, Xiangyu Zhang, and Xiaojuan Qi. Self-supervised visual representation learning with semantic grouping. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, 2022. 6

[34] Xin Wen, Bingchen Zhao, Anlin Zheng, Xiangyu Zhang, and XIAOJUAN QI. Self-supervised visual representation learning with semantic grouping. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Adv. Neural Inf. Process. Sys. (NeurIPS)*, 2022. 6, 7

[35] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 5, 7, 8

[36] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, pages 3733–3742, 2018. 2

[37] Tete Xiao, Colorado J Reed, Xiaolong Wang, Kurt Keutzer, and Trevor Darrell. Region similarity representation learning. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, pages 10539–10548, 2021. 2, 3, 6

[38] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In *IEEE/CVF Inter. Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 8392–8401, 2021. 6

[39] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *Adv. Neural Inf. Process. Sys. (NeurIPS)*, pages 16684–16693, 2021. 2, 3, 4, 6, 7

[40] Junwei Yang, Ke Zhang, Zhaolin Cui, Jinming Su, Junfeng Luo, and Xiaolin Wei. Inscon: instance consistency feature representation via self-supervised learning. *arXiv preprint arXiv:2203.07688*, 2022. 6

[41] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks, 2017. 8

[42] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Inter. Conf. Mach. Learn. (ICML)*, pages 12310–12320. PMLR, 2021. 2