A DATASETS AND EVALUATION DETAILS

We use different datasets throughout this work that we present below.

- For pretraining purposes, we use the standard *ImageNet ILSVRC 2012 (IN)* (Russakovsky et al., 2015) dataset, which contains 1.2M training images separated in 1000 class categories.
- We also use the *MS-COCO (COCO)* (Lin et al., 2014) dataset for pretraining, but mainly for finetuning and evaluation purposes. This dataset contains 80 classes of objects and about 118k training images *(train2017 subset)*. Performance is evaluated on the *val2017* subset.
- For finetuning, we also use the *Pascal VOC 2007 and 2012 (VOC 07-12)* (Everingham et al., 2010) datasets. This dataset contains 20 classes of objects, and we use the combination of the *trainval* subsets from both VOC2007 and VOC2012 for training, corresponding to about 16k training images in total. Performance is evaluated on the *test* subset from VOC2007.
- For a more complicated dataset, we use the *Few-Shot Object Detection* dataset (Fan et al., 2020). Since the dataset is designed as *open-set*, *i.e.* with different classes between training and testing, we separately use the *train* and *test* sets for benchmarking. We separate the test set into a training and testing subset, by randomly taking 80% of images for training and the 20% remaining for testing, and do the same for the train set. We make sure that all classes appears at least once in both training and testing subsets. The images selected for training and testing will be made available for reproducibility. This separation leads to about 11k training images and 3k testing images for the *test set*, and about 42k training images and 10k testing images for the *train* set.

B AUGMENTATIONS USED

We detail in Table 4 the distributions of augmentations used to create the weak view and the strong view. The *Weak Augmentations* follow standard supervised training for transformer-based detectors (Carion et al., 2020; Zhu et al., 2021). The *Strong Augmentations* follow typical contrastive learning augmentations (Chen et al., 2020a; Bar et al., 2022).

Table 4: The different sets of augmentations used for each branch (*weak* or *strong*). *Probability* indicates the probability of applying the corresponding augmentation.

Weak Augmentations ($\mathcal{T}^1$)		
Augmentations	Probability	Parameters
Horizontal Flip	0.5	–
Resize	0.5	*Mid-scale:* short edge = range(320,481,16) *Large-scale:* short edge = range(480,801,32)
Resize	0.5	short edge $\in \{400, 500, 600\}$
Random Size Crop		min size = 384 ; max size = 600
Resize		*Mid-scale:* short edge = range(320,481,16) *Large-scale:* short edge = range(480,801,32)
Strong Augmentations ($\mathcal{T}^2$)		
Color Jitter	0.8	(brightness, contrast, saturation, hue) = (0.4, 0.4, 0.4, 0.1)
GrayScale	0.2	–
Gaussian Blur	0.5	(sigma x, sigma y) = (0.1, 2.0)

C PRETRAINING COST

We compare the cost of pretraining *in terms of memory and hardware used* to SoCo (Wei et al., 2021) in Table 5 since it is the closest in terms of pretraining pipeline. The information are derived from their paper and official github repository.

Table 5: Comparison of pretraining cost between overall pretraining methods. We compare the number of pretraining epochs on IN, the total batch size, the total number of iterations, the total training time (in hours), the number of iterations per seconds (It. / sec.), and the hardware used (number and type of GPUs). We can see that our pretraining is globally less costly than SoCo.

Method	IN epochs	Batch Size	Iterations	Time	It. / sec.	Hardware
SoCo	400	2048	240k	140h	0.5	16 V100 32G
ProSeCo (Ours)	10	64	187k	40h	1.4	8 A100 40G

Even though A100 GPUs are faster than V100 GPUs, we are *training much faster* which is partly explained by the fact that they learn the backbone along with the detection heads during pretraining, leading to more parameters to learn and more computations. Furthermore, our ProSeCo requires a smaller batch size leading to less memory and thus less GPUs needed.

D USING ANOTHER CONTRASTIVE LOSS

The popular InfoNCE loss (Oord et al., 2018) used for contrastive learning is a similarity based function scaled by the temperature τ that maximizes agreement between the positive pair of instances and push negatives away. However, it suffers from the *class collision problem* (Cai et al., 2020; Denize et al., 2023), where semantically close instances can be used as negatives in the loss computation, which damages the quality of the representation learned. Recent work (Zheng et al., 2021; Denize et al., 2023) have tackled this problem by introducing the relational aspect between instances. In our experimental study, we also considered the InfoNCE loss for ProSeCo. We formulate the loss as follows:

$$\mathcal{L}_{\text{InfoNCE}}(\mathbf{z}, \hat{\mathbf{z}}, \hat{\sigma}^{\text{prop}}) = -\frac{1}{N_b N} \sum_{i=1}^{N_b} \sum_{j=1}^{N} \log \left(\frac{\exp(\mathbf{z}_{(i,j)} \cdot \hat{\mathbf{z}}_{(i,\hat{\sigma}_i^{\text{prop}}(j))}/\tau)}{\sum_{k=1}^{N_b} \sum_{l=1}^{N} \exp(\mathbf{z}_{(i,j)} \cdot \hat{\mathbf{z}}_{(k,l)}/\tau)} \right). \tag{11}$$

This formulation leads to an effective batch size of $N_b \cdot N$ for the contrastive loss. Similarly to our *LocSCE*, the localization of the objects can be introduced in the InfoNCE loss function to obtain the *LocNCE* objective as follows:

$$\mathcal{L}_{\text{LocNCE}}(\mathbf{y}, \hat{\mathbf{z}}, \hat{\sigma}^{\text{prop}}) = -\frac{1}{N_b N} \sum_{i=1}^{N_b} \sum_{j=1}^{N} \sum_{m=1}^{N} \mathbb{1}_{IoU_i(j,m) \geq \delta} \log \left(\frac{\exp(\mathbf{z}_{(i,j)} \cdot \hat{\mathbf{z}}_{(i,\hat{\sigma}_i^{\text{prop}}(m))}/\tau)}{\sum_{k=1}^{N_b} \sum_{l=1}^{N} \exp(\mathbf{z}_{(i,j)} \cdot \hat{\mathbf{z}}_{(k,l)}/\tau)} \right). \tag{12}$$

In Table 6, we compare the two different contrastive objectives for pretraining. We can see that using the InfoNCE loss leads to slightly better results (+0.3 p.p.). However, when using the localization information, SCE benefits much more than InfoNCE (+1.7 p.p. compared to +0.6 p.p.). This might be that the selection of positives from the localization information helps to introduce easy positive examples, and thanks to this, the relational aspect of SCE can focus on the more difficult positives. In the end, LocSCE achieves stronger results than LocNCE (+0.8 p.p.).

E FULL EVALUATION METRICS

Tables 7 and 8 provide the results with full evaluation metrics (mAP, AP_{50} and AP_{75} in %) on PASCAL VOC, Mini-VOC, FSOD-test and FSOD-train benchmarks.

Table 6: Performance (mAP in %) comparison on Mini-COCO 5% of the different contrastive loss and the effect of the localization information on each of them.

Loss	δ	mAP
InfoNCE	1.0	26.4
LocNCE (Ours)	0.5	**27.0**
SCE	1.0	26.1
LocSCE (Ours)	0.2	27.0
LocSCE (Ours)	0.7	27.1
LocSCE (Ours)	0.5	**27.8**

Table 7: Performance comparison after finetuning on PASCAL VOC and in the novel Mini-VOC setting. On Mini-VOC, we use different percentage of training data (with the corresponding number of images reported) for finetuning.

Method	PASCAL VOC			Mini-VOC					
	100% (16k)			5% (0.8k)			10% (1.6k)		
	mAP	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}
Supervised	59.5	82.6	65.6	33.9	56.9	35.0	40.8	63.7	43.1
DETReg (Bar et al., 2022)	63.5	83.3	70.3	43.1	63.4	46.1	48.2	68.6	51.9
ProSeCo (Ours)	**65.1**	**84.7**	**73.0**	**46.1**	**66.1**	**50.2**	**51.3**	**72.7**	**56.1**

F Increasing the number of queries

In Table 9, we provide an ablation on the number of object proposals (queries) N in Def. DETR, when pretraining with ProSeCo and finetuning afterward. A higher N leads to more parameters in the model and longer computing time, but we can see that the results of Def. DETR are relatively stable w.r.t. to the number of queries. On the other hand, ProSeCo benefits from increasing the number of queries, since it means a higher effective batch size during contrastive learning. However, the default value of $N = 300$ leads to the best results, both with and without pretraining.

G Finetuning with a lot of data

In Table 10, we present results when finetuning on the full COCO dataset under the $1\times$ training schedule (Wei et al., 2021; Li et al., 2022), *i.e.* 12 training epochs and decaying the learning rate in the last epoch. The improvements in the large-scale annotated data regime are limited, which can be observed also in previous work (Dai et al., 2021b; Bar et al., 2022). As we can see, our ProSeCo reaches similar results than DETReg (Bar et al., 2022). We believe that this limitation comes from the pretrained backbone that stays fixed during pretraining, and from the extensive supervision during fine-tuning. However, as we can see in both Tables 1b and 8, we outperform DETReg on our *FSOD-train* benchmark, which represents a setting with mid-scale annotated data (42k training images).

H Performance comparison of Detectors in the few data regime

From the results presented in Table 11, we can see that Def. DETR, a recent state-of-the-art detection model based on transformers, achieves consistently better performance than the most popular two-stage method in when learning with limited labels. These differences in performance are all the more impressive since the two methods have a similar number of parameters. These results motivated our choice of transformer-based architectures for our pretraining method.

Table 8: Performance comparison on FSOD-test and FSOD-train

Method	FSOD-test (11k)			FSOD-train (42k)		
	mAP	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}
Supervised	39.3	57.7	42.6	42.6	58.1	46.5
DETReg (Bar et al., 2022)	43.2	59.6	47.8	43.3	57.9	47.3
ProSeCo (Ours)	**46.6**	**64.5**	**50.9**	**47.2**	**62.4**	**51.7**

Table 9: Performance (mAP in %) comparison on Mini-COCO 5% when changing the number of object proposals in Def. DETR.

Method	N	Performance
Supervised	100	23.1
	200	23.0
	300	**23.6**
	500	23.3
ProSeCo (Ours)	100	25.7
	200	26.5
	300	**27.8**
	500	27.2

Table 10: Performance (mAP, AP_{50} and AP_{75} in %) comparison on the full COCO dataset (118k training images) with the $1\times$ training schedule.

Method	COCO (118k)		
	mAP	AP_{50}	AP_{75}
Supervised	37.4	55.5	40.5
DETReg (Bar et al., 2022)	38.9	56.6	42.3
ProSeCo (Ours)	38.9	56.2	42.4

Table 11: Performance (mAP in %) comparison between Faster-RCNN (FRCNN) (Ren et al., 2015) with Feature Pyramid Network (FPN) (Lin et al., 2017), a two-stage detector commonly used, FCOS (Tian et al., 2019), a more recent one-stage method, and Deformable DETR (Def. DETR) (Zhu et al., 2021), a state-of-the-art transformer-based object detector, with the same ResNet-50 backbone model.
† Results from Liu et al. (2022b). ‡ Results from Bouniot et al. (2023).

Method	Mini-COCO			
	0.5% (590)	1% (1.2k)	5% (5.9k)	10% (11.8k)
FCOS†	5.42 ± 0.01	8.43 ± 0.03	17.01 ± 0.01	20.98 ± 0.01
FRCNN + FPN†	6.83 ± 0.15	9.05 ± 0.16	18.47 ± 0.22	23.86 ± 0.81
Def. DETR‡	$\mathbf{8.95 \pm 0.51}$	$\mathbf{12.96 \pm 0.08}$	$\mathbf{23.59 \pm 0.21}$	$\mathbf{28.55 \pm 0.08}$