Supplementary Material

The hyper-parameter settings are introduced first. Then we introduce the architectures of projection heads used in SCFS in Fig. S1. And the Pytorch-style pseudocode and training time of SCFS are introduced. Finally, we show more visualization results for the attention maps of SCFS between local images and corresponding global images in Fig. S2 and Fig. S3.

A. Hyper-parameters Setting

During the pretraining procedure, we follow the most hyper-parameters setting of DINO [4]. The SGD optimizer is used and the learning rate is linearly warmed up to its base value during the first 10 epochs. The base learning rate is set according to the linear scaling rule: $lr = 0.1 \times batchsize/256$. After the warm-up procedure, the learning rate is decayed with a cosine schedule [27]. The weight decay is set to $1e-4$. For the temperatures, τ is set to 0.1, and a linear warm-up from 0.04 to 0.07 is set to τ' during the first 50 epochs. Following DINO [4], the centering operation is applied to the output of the momentum encoder to avoid collapse. For data augmentation, the global augmentations consist of random cropping (with a scale of 0.14-1), resizing to 224×224, random horizontal flip, gaussian blur, and color jittering. And the local augmentations consist of random cropping (with a scale of 0.05-0.14 by default), resizing to 96×96, random horizontal flip, gaussian blur, and color jittering. 2 global views with $N = 8$ local views are the default setting of augmentation.

During the linear probing procedure, we evaluate the representation quality with a linear classifier. The linear classifier is trained with the SGD optimizer and a batch size of 1024 for 100 epochs on ImageNet. Weight decay is not used. For data augmentation, only random resizes crops and horizontal flips are applied.

B. Projection Head

There are two kinds of projection heads in SCFS. The projection head for the contrast between data augmentations consists of a four-layer MLP with the same architecture as DINO [4]. As shown in Fig. S1 (a), the hidden layers are with 2048 dimension and are with gaussian error linear units (GELU) activations. After the MLP, a L_2 normalization and a weight normalized FC layer with K ($K = 65536$) dimension are applied.

The projection head for feature search consists of three convolutional layers and two FC layers. The detailed architecture is shown in Fig. S1 (b). To make the feature search loss easy to backward, the residual connection is applied to the three convolutional layers. After global-averaged pooling, two FC layers are applied to project features to the output dimension. Note that the output dimension is set to 256, which achieves good performance in all the experiments.

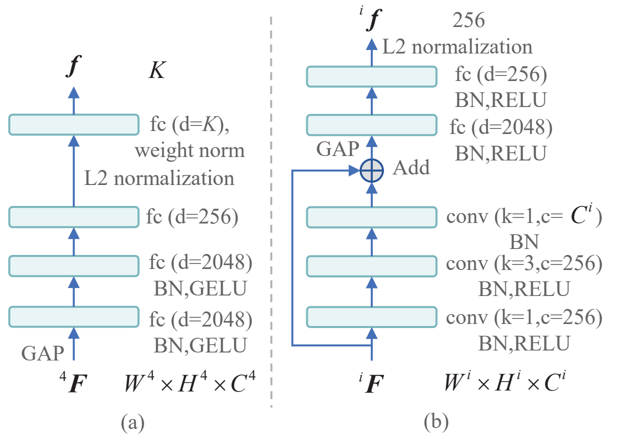

Figure S1. Architecture of the projection heads in SCFS. (a) projection head for the contrast between data augmentations; (b) projection head for feature search.

Method	Batch Size	Epochs	Time	Mem.
DINO [4]	256	200	147h	52.0G
SCFS	256	200	192h	68.8G

Table S1. Training Time and Memory Requirements.

C. PyTorch-Style Pseudocode

The Pytorch-style pseudocode of SCFS is shown in algorithm 1. For simplification, we only show one local augmentation and the i-th layer for feature search.

D. Training Time and Memory Requirements

We test the training times and memory requirements on a machine with 8 NVIDIA GeForce RTX 2080Ti GPUs. As shown in Tab. S1, compared to the baseline DINO [4], the extra computational time of SCFS increases by 30%. The memory requirement of SCFS increases 32% compared to DINO [4]. But this memory requirement increase doesn't constrain batch size too much. And SCFS achieves better performance with a smaller batch size compared to DINO.

E. Results of Other Backbones on ImageNet

We extend the experiments in Tab. 10 to compare with other methods for different backbones on ImageNet. As shown in Tab. S2, SCFS achieves the best performance on ImageNet with the ViT-S and ViT-B backbones.

F. Results on ImageNet-21k

We further pre-train SCFS and DINO [4] on ImageNet-21k. They are pre-trained with 1024 batch size for 20 epochs on ImageNet-21k. We evaluate the k-NN top-1 accuracy on

Algorithm 1 PyTorch-style pseudocode of SCFS.

```
# es, et: encoder and momentum encoder networks
# hs_i, ht_i: head on the layer-i for feature search of
  the encoder and momentum encoder
# C, Ci: centers
# tps, tpt: temperatures
# l, m: network and center momentum rates
et.params = es.params
for I in loader: # load a minibatch I with n samples
    I1, I2 = augment(I), augment(I) # global views
    Il = augment(I) # multiple local views

    # encoder output
    s1, _, = es(I1)
    s2, _, = es(I2)
    sl, Sl_i = es(Il)

    # momentum encoder output
    t1, T1_i = es(I1)
    t2, T2_i = es(I2)

    # feature search
    sl_i_1, t1_i = FS(Sl_i, T1_i, hs_i, ht_i)
    sl_i_2, t2_i = FS(Sl_i, T2_i, hs_i, ht_i)

    # contrastive loss for data augmentation
    loss_g = H(t1, s2, C)/2 + H(t2, s1, C)/2
    loss_l = H(t1, sl, C)/2 + H(t2, sl, C)/2
    loss_d = loss_g + loss_l

    # feature search loss
    loss_fs = H(t1_i, sl_i_1, Ci)/2 + H(t2_i, sl_i_2, Ci
     )/2

    # total loss
    loss = loss_d + loss_fs
    loss.backward() # back-propagate

    # encoder, momentum encoder and center updates
    update(es) # SGD
    et.params = l*et.params + (1-l)*es.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)
    Ci = m*Ci + (1-m)*cat([t1_i, t2_i]).mean(dim=0)

def H(t, s, C):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()

def FS(t, s, hs, ht):
    t = t.detach() # stop gradient
    s = gap(s, dim=(1,2)) # gap
    s = normalize(s, dim=1) # l2-normalize
    t = normalize(t, dim=3) # l2-normalize
    a = (s*t).sum(dim=3) # similarity
    s = a*s
    return hs(s), ht(t)
```

ImageNet. As shown in Tab. S3, SCFS also performs better than DINO [4].

G. Comparing cross-view search with self-view attention

SCFS mitigates semantic ambiguity to improve contrastive learning by building cross-view search tasks, while some studies strengthen self-view representation by attention mechanisms such as VITO [30]. In this section, we discuss the effectiveness of the two approaches. We implemented VITO [30] based on DINO [4] to achieve a fair comparison. As shown in the Tab. S4, with the same training settings (pre-training on ImageNet-100, 256 batch size, 200epochs), both SCFS and VITO [30] perform better than DINO [4], which indicates semantic contrast consistency is crucial. In addition, SCFS achieves 1% performance improvement compared to VITO [30], which indicates that strengthening the cross-view semantic consistency is more effective for contrastive learning.

Method	Backbone	BS	Epoch	k-NN	LP
SimCLR [5]	ViT-S-16	4096	300	-	69.0
BYOL [13]	ViT-S-16	1024	300	66.6	71.4
SwAV [3]	ViT-S-16	1024	300	64.7	71.8
MoCo-v3 [9]	ViT-S-16	4096	300	-	72.5
DINO [4]	ViT-S-16	256	100	69.9	73.8
SCFS	ViT-S-16	256	100	**70.7**	**74.7**
MoCo-v3 [9]	ViT-B-16	4096	300	-	76.5
DINO [4]	ViT-B-16	256	100	73.6	77.0
SCFS	ViT-B-16	256	100	**75.8**	**78.0**

Table S2. Results of Other Backbones on ImageNet.

Method	k-NN
DINO [4]	48.6
SCFS	**50.0**

Table S3. Results on ImageNet-21k.

Method	k-NN	LP
DINO [4]	81.1	87.0
VITO [30]	84.1	88.2
SCFS	**84.8**	**89.2**

Table S4. Comparison results of DINO, VITO and SCFS.

H. More Visualization Results

We visualize the attention maps of SCFS between local images and corresponding global image. As shown in Fig. S2, SCFS can accurately focus on semantics-consistent regions between global images and local images. According to the different semantic concepts inputs, consistent semantic information can be searched on the global feature.

Furthermore, we also visualize the attention maps between local images and another images that contains objects with the same category. As shown in Fig. S3, the attention maps show that the semantics-consistent regions between different images are also activated. When the background images are input, the global images are no longer activated incorrectly, which achieves the contrastive noise mitigation and demonstrates the effectiveness of SCFS.

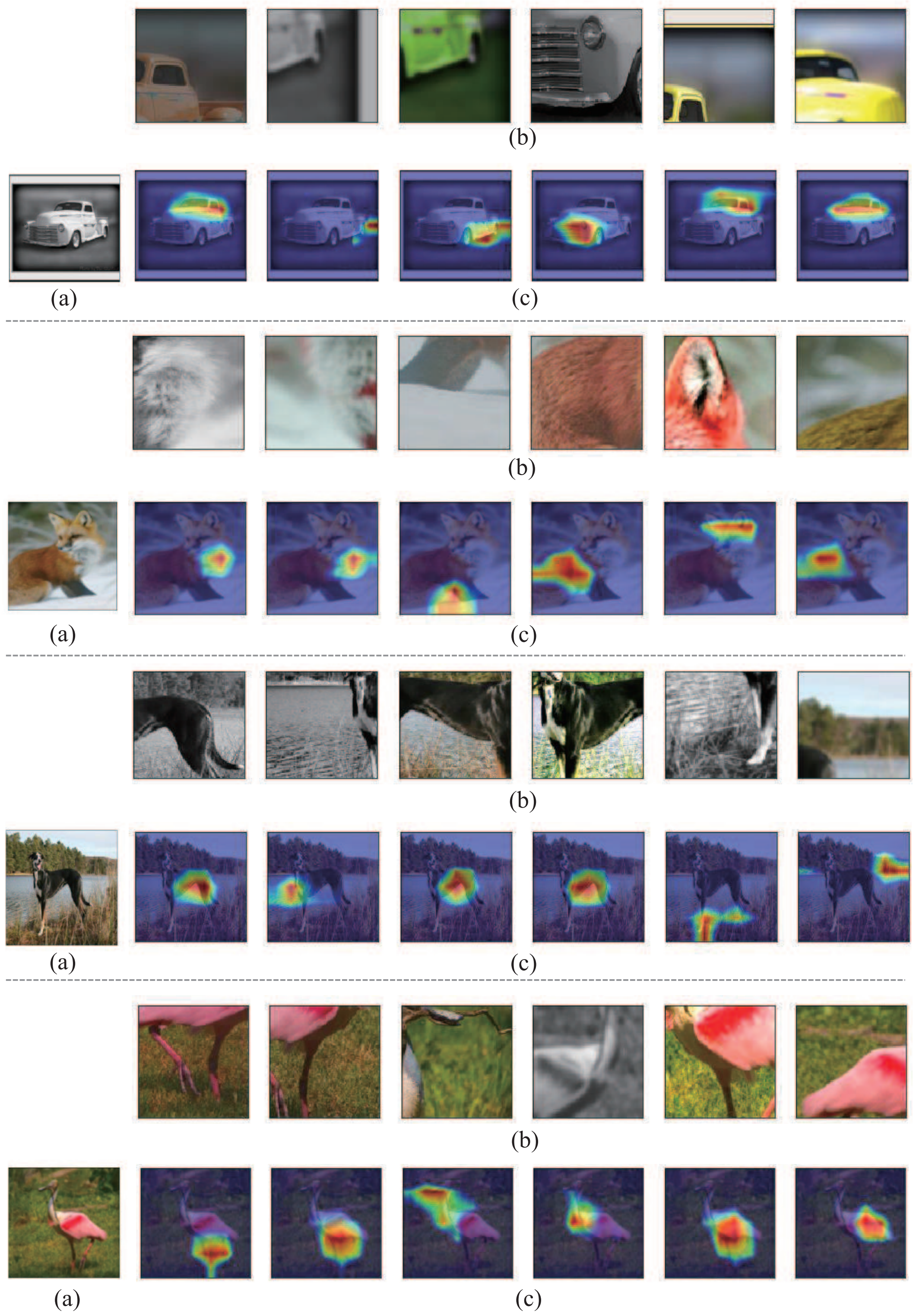

Figure S2. Attention maps of SCFS between local images and corresponding global image. In each example, (a) shows a global image, (b) shows six local augmentations of the global image, and (c) shows the attention maps that highlight the semantics-consistent regions between the local images in (b) and the global image in (a), which are obtained by multiplying the globally average pooled feature maps from the encoder (Res4) of the local images in (b) with the feature map (Res4) of the global image in (a).

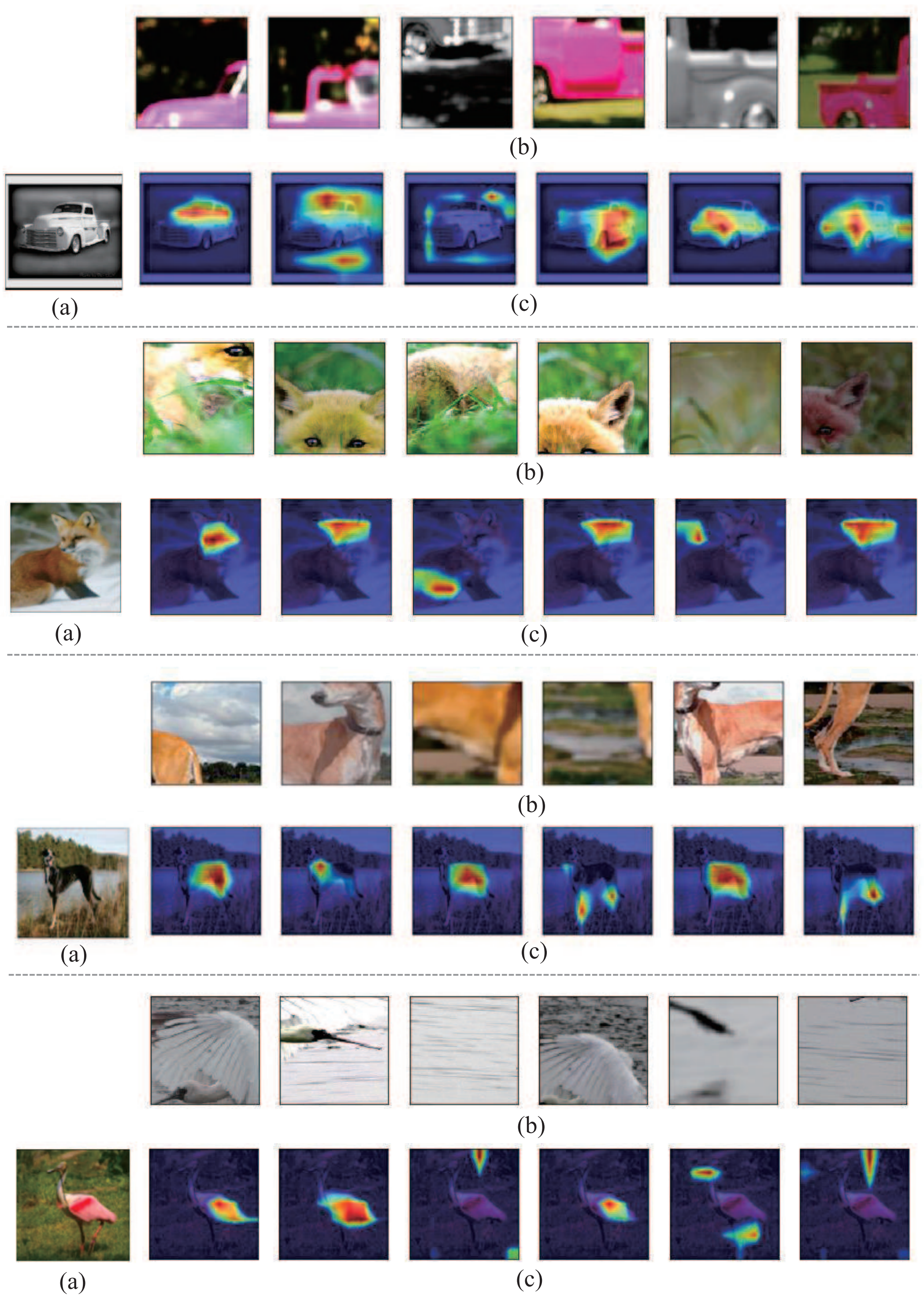

Figure S3. Attention maps of SCFS between local images and another image that contains objects with the same category. In each example, (a) shows an image that contains objects with the same category in (b), (b) shows six local augmentations of a global image, and (c) shows the attention maps that highlight the semantics-consistent regions between the local images in (b) and the image in (a), which are obtained by multiplying the globally average pooled feature maps from the encoder (Res4) of the local images in (b) with the feature map (Res4) of the image in (a).