Similarity Contrastive Estimation for Image and Video Soft Contrastive Self-Supervised Learning

Julien Denize[1,2*], Jaonary Rabarisoa[1†], Astrid Orcesi[1†] and Romain Hérault[2†]

[1*]LVA, CEA LIST, Université Paris-Saclay, Palaiseau, F-91120, France.
[2*]LITIS, INSA Rouen, Normandie Univ, Saint Etienne du Rouvray, 76801, France.

*Corresponding author. E-mail: julien.denize@cea.fr;
Contributing authors: jaonary.rabarisoa@cea.fr; astrid.orcesi@cea.fr;
romain.herault@insa-rouen.fr;
†These authors contributed equally to this work.

A Classes to construct ImageNet100

To build the ImageNet100 dataset, we used the classes shared by the CMC [1] authors in the supplementary material of their publication. We also share these classes in Tab. 1.

100 selected classes from ImageNet			
n02869837	n01749939	n02488291	n02107142
n13037406	n02091831	n04517823	n04589890
n03062245	n01773797	n01735189	n07831146
n07753275	n03085013	n04485082	n02105505
n01983481	n02788148	n03530642	n04435653
n02086910	n02859443	n13040303	n03594734
n02085620	n02099849	n01558993	n04493381
n02109047	n04111531	n02877765	n04429376
n02009229	n01978455	n02106550	n01820546
n01692333	n07714571	n02974003	n02114855
n03785016	n03764736	n03775546	n02087046
n07836838	n04099969	n04592741	n03891251
n02701002	n03379051	n02259212	n07715103
n03947888	n04026417	n02326432	n03637318
n01980166	n02113799	n02086240	n03903868
n02483362	n04127249	n02089973	n03017168
n02093428	n02804414	n02396427	n04418357
n02172182	n01729322	n02113978	n03787032
n02089867	n02119022	n03777754	n04238763
n02231487	n03032252	n02138441	n02104029
n03837869	n03494278	n04136333	n03794056
n03492542	n02018207	n04067472	n03930630
n03584829	n02123045	n04229816	n02100583
n03642806	n04336792	n03259280	n02116738
n02108089	n03424325	n01855672	n02090622

Table 1: The 100 classes selected from ImageNet to construct ImageNet100.

B Pseudo-Code of SCE

```
# dataloader: loader of batches
# bsz: batch size
# epochs: number of epochs
# T1: weak distribution of data augmentations
# T2: strong distribution of data augmentations
# f_s, g_s, h_s: online encoder, projector, and optional predictor
# f_t, g_t: momentum encoder and projector
# queue: memory buffer
# tau: online temperature
# tau_m: momentum temperature
# lambda_: coefficient between contrastive and relational aspects
# symmetry_loss: if True, symmetries the loss

def sce_loss(z1, z2):
    sim2_pos = zeros(bsz)
    sim2_neg = einsum("nc,kc->nk", z2, queue)
    sim2 = cat([sim2_pos, sim2_neg]) / tau_m
    s2 = softmax(sim2)
    w2 = lambda_ * one_hot(sim2_pos, bsz+1) + (1 - lambda_) * s

    sim1_pos = einsum("nc,nc->n", z1, z2)
    sim1_neg = einsum("nc,kc->nk", z1, queue)
    sim1 = cat([sim1_pos, sim1_neg]) / tau
    p1 = softmax(sim1)

    loss = cross_entropy(p1, w2)
    return loss

for i in range(epochs):
    for x in dataloader:
        x1, x2 = T1(x), T2(x)

        z1_s, z2_t = h_s(g_s(f_s(x1))), g_t(f_t(x2))
        z2_t = stop_grad(z2_t)

        loss = sce_loss(z1_s, z2_t)
        if symmetry_loss:
            z1_t, z2_s = g_t(f_t(x1)), h_s(g_s(f_s(x2)))
            z1_t = stop_grad(z1_t)
            loss += sce_loss(z2_s, z1_t)
            loss /= 2
        loss.backward()

        update(f_s.params)
        update(g_s.params)
        update(h_s.params)
        momentum_update(f_t.params, f_s.params)
        momentum_update(g_t.params, g_s.params)

        fifo_update(queue, z2_t)
        if symmetry_loss:
            fifo_update(queue, z1_t)
```

Algorithm 1: Pseudo-Code of SCE in Pytorch style

C Proof Proposition 1.

Proposition. *L_{SCE} defined as*

$$L_{SCE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N} w_{ik}^2 \log\left(p_{ik}^1\right),$$

can be written as:

$$L_{SCE} = \lambda \cdot L_{InfoNCE} + \mu \cdot L_{ReSSL} + \eta \cdot L_{ceil},$$

with $\mu = \eta = 1 - \lambda$ and

$$L_{Ceil} = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right).$$

Proof Recall that:

$$\begin{aligned}
p_{ik}^1 &= \frac{\exp(\mathbf{z_i^1} \cdot \mathbf{z_k^2}/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)},\\
s_{ik}^2 &= \frac{\mathbb{1}_{i\neq k} \cdot \exp(\mathbf{z_i^2} \cdot \mathbf{z_k^2}/\tau_m)}{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^2} \cdot \mathbf{z_j^2}/\tau_m)},\\
w_{ik}^2 &= \lambda \cdot \mathbb{1}_{i=k} + (1-\lambda) \cdot s_{ik}^2.
\end{aligned}$$

We decompose the second loss over k in the definition of L_{SCE} to make the proof:

$$\begin{aligned}
L_{SCE} &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N} w_{ik}^2 \log\left(p_{ik}^1\right)\\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[w_{ii}^2 \log\left(p_{ii}^1\right) + \sum_{\substack{k=1\\k\neq i}}^{N} w_{ik}^2 log\left(p_{ik}^1\right)\right]\\
&= \underbrace{-\frac{1}{N}\sum_{i=1}^{N} w_{ii}^2\left(p_{ii}^1\right)}_{(1)} \underbrace{-\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N} w_{ik}^2 \log\left(p_{ik}^1\right)}_{(2)}.
\end{aligned}$$

First we rewrite (1) to retrieve the $L_{InfoNCE}$ loss.

$$\begin{aligned}
(1) &= -\frac{1}{N}\sum_{i=1}^{N} w_{ii}^2 log\left(p_{ii}^1\right)\\
&= -\frac{1}{N}\sum_{i=1}^{N} \lambda \cdot \log\left(p_{ii}^1\right)\\
&= -\lambda \cdot \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{\exp(\mathbf{z_i^1} \cdot \mathbf{z_i^2}/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right)\\
&= \lambda \cdot L_{InfoNCE}.
\end{aligned}$$

Now we rewrite (2) to retrieve the L_{ReSSL} and L_{Ceil} losses.

$$\begin{aligned}
(2) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N} w_{ik}^2 \log\left(p_{ik}^1\right)\\
&= -\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N} (1-\lambda) \cdot s_{ik}^2 \cdot \log\left(p_{ik}^1\right)\\
&= -(1-\lambda) \cdot \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N} s_{ik}^2 \cdot \log\left(p_{ik}^1\right)\\
&= -(1-\lambda) \cdot \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N}\left[s_{ik}^2 \cdot \log\left(\frac{\exp(\mathbf{z_i^1} \cdot \mathbf{z_k^2}/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right)\right]\\
&= -(1-\lambda) \cdot \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N}\Bigg[s_{ik}^2 \cdot \Bigg(\\
&\qquad \log\left(\exp(\mathbf{z_i^1} \cdot \mathbf{z_k^2}/\tau)\right) - \log\left(\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)\right)\Bigg)\Bigg]\\
&= -(1-\lambda) \cdot \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N}\Bigg[s_{ik}^2 \cdot \Bigg(\\
&\qquad \log\left(\exp(\mathbf{z_i^1} \cdot \mathbf{z_k^2}/\tau)\right) - \log\left(\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)\right) +\\
&\qquad \log\left(\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)\right) -\\
&\qquad \log\left(\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)\right)\Bigg)\Bigg]\\
&= -(1-\lambda) \cdot \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N}\Bigg[s_{ik}^2 \cdot \Bigg(\\
&\qquad \log\left(\frac{\exp(\mathbf{z_i^1} \cdot \mathbf{z_k^2}/\tau)}{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right) +\\
&\qquad \log\left(\frac{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right)\Bigg)\Bigg]\\
&= -(1-\lambda) \cdot \frac{1}{N}\Bigg(\\
&\sum_{i=1}^{N}\sum_{k=1}^{N}\left[s_{ik}^2 \cdot \log\left(\frac{\exp(\mathbf{z_i^1} \cdot \mathbf{z_k^2}/\tau)}{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right)\right] +\\
&\sum_{i=1}^{N}\sum_{k=1}^{N}\left[s_{ik}^2 \cdot \log\left(\frac{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right)\right]\Bigg).
\end{aligned}$$

Because $s_{ii}^2 = 0$ and $\mathbf{s_i^2}$ is a probability distribution, we have:

$$\begin{aligned}
(2) &= -(1-\lambda) \cdot\\
&\qquad \frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N}\left[s_{ik}^2 \cdot \log\left(\frac{\mathbb{1}_{i\neq k} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_k^2}/\tau)}{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right)\right] -\\
&\qquad (1-\lambda) \cdot \frac{1}{N}\sum_{i=1}^{N}\left[\log\left(\frac{\sum_{j=1}^{N} \mathbb{1}_{i\neq j} \cdot \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{z_i^1} \cdot \mathbf{z_j^2}/\tau)}\right)\right]\\
&= (1-\lambda) \cdot L_{ReSSL} + (1-\lambda) \cdot L_{Ceil}.
\end{aligned}$$

□

Backbone	Dataset	Projector				Input	Buffer	ema	LR	Batch	WD
		Layers	Hid dim	Out dim	BN						
R-18	CIFAR	2	512	128	hid	32^2	4,096	0.900	0.06	256	$5e^{-4}$
R-18	STL10	2	512	128	hid	96^2	16,384	0.996	0.06	256	$5e^{-4}$
R-18	Tiny-IN	2	512	128	hid	64^2	16,384	0.996	0.06	256	$5e^{-4}$
R-50	IN100	2	4096	256	no	224^2	65,536	0.996	0.3	512	$1e^{-4}$
R-50	IN1k	3	2048	256	all	224^2	65,536	0.996	0.5	512	$1e^{-4}$

Table 2: Architecture and hyperparameters used for pretraining on the different datasets. **LR** stands for the initial learning rate, **WD** for weight decay, **BN** for batch normalization [2], **Hid** for hidden, **Dim** for dimension, **ema** for the initial momentum value used to update the momentum branch. For **BN**: "no" means no batch normalization is used in the projector, "hid" means batch normalization after each hidden layer, "all" means batch normalization after the hidden layer and the output layer.

D Implementation details

D.1 Ablation study and baseline comparison for images

Pretraining Implementation details. We use the ResNet-50 [3] encoder for large datasets and ResNet-18 for small and medium datasets with changes detailed below. We pretrain the models for 200 epochs. We apply by default *strong* and *weak* data augmentations, defined in Tab. 1 in the main paper, with the scaling range for the random resized crop set to $(0.2, 1.0)$. Specific hyperparameters for each dataset for the projector construction, the size of the input, the size of the memory buffer, the initial momentum value, the initial learning rate, the batch size and the weight decay applied can be found in Tab. 2. We use the SGD optimizer [4] with a momentum of 0.9. A linear warmup is applied during 5 epochs to reach the initial learning rate. The learning rate is scaled using the linear scaling rule and follows the cosine decay scheduler without restart [5]. The momentum value to update the target branch follows a cosine strategy from its initial value to reach 1 at the end of training. We do not symmetrize the loss by default.

Architecture change for small and medium datasets. Because the images are smaller, and ResNet is suitable for larger images, typically 224×224, we follow guidance from SimCLR [6] and replace the first 7×7 Conv of stride 2 with a 3×3 Conv of stride 1. We also remove the first pooling layer.

Evaluation protocol. To evaluate our pretrained encoders, we train a linear classifier following [7, 8]. We train for 100 epochs on top of the frozen pretrained encoder using an SGD optimizer with an initial learning rate of 30 without weight decay and a momentum of 0.9. A scheduler is applied to the learning rate that is decayed by a factor of 0.1 at 60 and 80 epochs. The data augmentations for the different datasets are:

- **training set for large datasets**: random resized crop to resolution 224×224 with the scaling range set to $(0.08, 1.0)$ and a random horizontal flip with a probability of 0.5.
- **training set for small and medium datasets**: random resized crop to the dataset resolution with a padding of 4 for small datasets and the scaling range set to $(0.08, 1.0)$. Also, a random horizontal flip with a probability of 0.5 is applied.
- **validation set for large datasets**: resize to resolution 256×256 and center crop to resolution 224×224.
- **validation set for small and medium datasets**: resize to the dataset resolution.

D.2 Imagenet study

Pretraining implementation details. We use the ResNet-50 [3] encoder and apply *strong-*α and *strong-*β augmentations, defined in Tab. 1 in the main paper, with the scaling range for the random resized crop set to $(0.2, 1.0)$. The batch size is set to 4096 and the memory buffer to 65,536. We follow the same training hyperparameters as [9] for the architecture. Specifically, we use the same projector and predictor, the LARS optimizer [10]

with a weight decay of $1.5 \cdot 10^{-6}$ for 1000 epochs of training and 10^{-6} for fewer epochs. Bias and batch normalization [2] parameters are excluded. The initial learning rate is 0.5 for 100 epochs and 0.3 for more epochs. It is linearly scaled for 10 epochs and it follows the cosine annealed scheduler. The momentum value follows a cosine scheduler from 0.996 for 1000 epochs, 0.99 for fewer epochs, to reach 1 at the end of training. The loss is symmetrized. For SCE specific hyperparameters, we keep the best from ablation study: $\lambda = 0.5$, $\tau = 0.1$ and $\tau_m = 0.07$.

Multi-crop setting. We follow Hu et al. [11] and sample 6 different views. The first two views are global views as without multi-crop, meaning resolution of 224×224 and the scaling range for random resized crop set to $(0.2, 1.0)$. The 4 local crops have a resolution of 192×192, 160×160, 128×128, 96×96 and scaling range (0.172, 0.86), (0.143, 0.715), (0.114, 0.571), (0.086, 0.429) on which we apply the *strong-*γ data augmentation defined in Tab. 1 in the main paper.

Evaluation protocol. We follow the protocol defined by [9]. Specifically, we train a linear classifier for 90 epochs on top of the frozen encoder with a batch size of 1024 and a SGD optimizer with a momentum of 0.9 and without weight decay. The initial learning rate is 0.1 and scaled using the linear scaling rule and follows the cosine decay scheduler without restart [5]. The data augmentations applied are:

- **training set**: random resized crop to resolution 224×224 with the scaling range set to $(0.08, 1.0)$ and a random horizontal flip with a probability of 0.5.
- **validation set**: resize to resolution 256×256 and center crop to resolution 224×224.

D.3 Video study

Pretraining implementation details. We used the ResNet3D-18 and ResNet3D-50 networks [12] following the Slow path of Feichtenhofer et al. [13]. The exact architecture details can be found in Tab. 3. We kept the siamese architecture used for ImageNet in Sec. 4.1.3 and depending on the backbone and pretraining dataset, the projector and predictor architectures as well as the memory buffer size vary and are referenced in Tab. 4. The LARS optimizer with a weight decay of 1.10^{-6}, batch normalization and bias parameters excluded, for 200 epoch of training is used. The learning rate follows a linear warmup until it reaches an initial value of 2.4 and then follows a cosine annealed scheduler. The initial learning rate is scaled following the linear scaling rule and the batch size is set to 512. The momentum value follows a cosine scheduler from 0.99 to 1 and the loss is symmetrized. To sample and crop different views from a video, we follow Feichtenhofer et al. [14] and sample randomly different clips from the video that lasts 2.56 seconds. For Kinetics it corresponds to 64 frames for a frame rate per second (FPS) of 25. Out of this clip we keep a number of frames specified in the main paper. By default, we sample two different clips to form positives and we apply the *strong-*α and *strong-*β augmentations, defined in Tab. 1 in the main paper, to the views.

stage	ResNet3d-18	ResNet3D-50
conv1	$1 \times 7^2, 64$ stride $1, 2^2$	$1 \times 7^2, 64$ stride $1, 2^2$
pool1	$1 \times 3^2, max$ stride $1, 2^2$	$1 \times 3^2, max$ stride $1, 2^2$
res_2	$\begin{bmatrix} 1 \times 3^2, 64 \\ 1 \times 3^2, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$
res_3	$\begin{bmatrix} 1 \times 3^2, 128 \\ 1 \times 3^2, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$
res_4	$\begin{bmatrix} 3 \times 3^2, 256 \\ 1 \times 3^2, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$
res_5	$\begin{bmatrix} 3 \times 3^2, 512 \\ 1 \times 3^2, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$
pool	global average	global average

Table 3: ResNet3D-18 and ResNet3D-50 networks.

Linear evaluation protocol details. We follow Feichtenhofer et al. [14] and train a linear classifier for 60 epochs on top of the frozen encoder with a batch size of 512. We use the SGD optimizer with a momentum of 0.9 and without weight decay to reach the initial learning rate 2 that follows the

Backbone	Dataset	Projector				Predictor				Buffer
		Layers	Hid dim	Out dim	BN	Layers	Hid dim	Out dim	BN	
ResNet3D-18	K200	3	1024	256	all	2	1024	256	hid	32768
ResNet3D-18	K400	3	1024	256	all	2	1024	256	hid	65536
ResNet3D-50	K400	3	4096	256	all	2	4096	256	hid	65536

Table 4: Architecture and hyperparameters used for video pretraining. **BN** stands for for Batch Normalization, **Hid** for hidden, **Dim** for dimension. For **BN**: "hid" means batch normalization after each hidden layer, "all" means batch normalization after the hidden layer and the output layer.

Dataset	τ	$\tau_m = 0.03$	$\tau_m = 0.04$	$\tau_m = 0.05$	$\tau_m = 0.06$	$\tau_m = 0.07$	$\tau_m = 0.08$	$\tau_m = 0.09$	$\tau_m = 0.1$
CIFAR10	0.1	89.93	90.03	90.06	90.20	90.16	90.06	89.67	88.97
CIFAR10	0.2	89.98	90.12	90.12	90.05	90.13	90.09	90.22	**90.34**
CIFAR100	0.1	64.49	64.90	65.19	65.33	65.27	**65.45**	64.89	63.87
CIFAR100	0.2	63.71	63.74	63.89	64.05	64.24	64.23	64.10	64.30
STL10	0.1	89.34	**89.94**	89.87	89.84	89.72	89.52	88.99	88.41
STL10	0.2	88.4	88.23	88.4	88.35	87.54	88.32	88.80	88.59
Tiny-IN	0.1	50.23	51.12	51.41	51.66	**51.90**	51.58	51.37	50.46
Tiny-IN	0.2	48.56	48.85	48.35	48.98	49.06	49.15	49.66	49.64

Table 5: Effect of varying the temperature parameters τ_m and τ on the Top-1 accuracy.

linear scaling rule with the batch size set to 512. A linear warmup is applied during 35 epochs and then a cosine annealing scheduler. For training, we sample randomly a clip in the video and random crop to the size 224×224 after short scaling the video to 256. An horizontal flip is also applied with a probability of 0.5. For evaluation, we follow the standard evaluation protocol of Feichtenhofer et al. [13] and sample 10 temporal clips with 3 different spatial crops of size 256×256 applied to each temporal clip to cover the whole video. The final prediction is the mean average of the predictions of the 30 clips sampled.

Finetuning evaluation protocol details. We follow Feichtenhofer et al. [14] for finetuning on UCF101 and HMDB51. We finetune the whole pretrained network and perform supervised training on the 101 and 51 classes respectively for 200 epochs with dropout of probability 0.8 before classification. We use the SGD optimizer with a momentum of 0.9 and without weight decay to reach the initial learning rate 0.1 that follows the linear scaling rule with the batch size set to 64 and a cosine annealing scheduler without warmup. For training, we sample randomly a clip in the video and random crop to the size 224×224 after short scaling the video to 256. We apply color jittering with the *strong* augmentation parameters, defined in Tab. 1 in the main paper, and an horizontal flip with a probability of 0.5. For evaluation, we follow the 30-crops procedure as for linear evaluation. Specific hyperparameter search for each dataset might improve results.

E Temperature influence on small and medium datasets

We made a temperature search on CIFAR10, CIFAR100, STL10 and Tiny-ImageNet by varying τ in $\{0.1, 0.2\}$ and τ_m in $\{0.03, ..., 0.10\}$. The results are in Tab. 5. As for ImageNet100, we need a sharper distribution on the output of the momentum encoder. Unlike ReSSL [8], SCE do not collapse when $\tau_m \to \tau$ thanks to the contrastive aspect. For our baselines comparison in Sec. 4.2, we use the best temperatures found for each dataset.

References

[1] Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: 16th European Conference on Computer Vision,

pp. 776–794 (2020). https://doi.org/10.1007/978-3-030-58621-8_45

[2] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: 32nd International Conference on Machine Learning, pp. 448–456 (2015). https://doi.org/10.1109/CVPR46437.2021.00113

[3] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

[4] Sutskever, I., Martens, J., Dahl, G.E., Hinton, G.E.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning, pp. 1139–1147 (2013)

[5] Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. 5th International Conference on Learning Representations (2017) https://arxiv.org/abs/1608.03983

[6] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th International Conference on Machine Learning, pp. 1597–1607 (2020)

[7] Chen, X., Fan, H., Girshick, R.B., He, K.: Improved baselines with momentum contrastive learning. arXiv **abs/2003.04297** (2020)

[8] Zheng, M., You, S., Wang, F., Qian, C., Zhang, C., Wang, X., Xu, C.: Ressl: Relational self-supervised learning with weak augmentation. In: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, pp. 2543–2555 (2021)

[9] Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: International Conference on Computer Vision, pp. 9620–9629 (2021). https://doi.org/10.1109/ICCV48922.2021.00950

[10] You, Y., Gitman, I., Ginsburg, B.: Large batch training of convolutional networks. arXiv **abs/1708.03888** (2017)

[11] Hu, Q., Wang, X., Hu, W., Qi, G.: Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In: Conference on Computer Vision and Pattern Recognition, pp. 1074–1083 (2021)

[12] Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: Conference on Computer Vision and Pattern Recognition, pp. 6546–6555 (2018). https://doi.org/10.1109/CVPR.2018.00685

[13] Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: International Conference on Computer Vision, pp. 6201–6210 (2019). https://doi.org/10.1109/ICCV.2019.00630

[14] Feichtenhofer, C., Fan, H., Xiong, B., Girshick, R.B., He, K.: A large-scale study on unsupervised spatiotemporal representation learning. In: Conference on Computer Vision and Pattern Recognition, pp. 3299–3309 (2021). https://doi.org/10.1109/CVPR46437.2021.00331