Constrained Mean Shift Using Distant Yet Related Neighbors for Representation Learning: Supplementary Material

K L Navaneet[1] *, Soroush Abbasi Koohpayegani[1] *, Ajinkya Tejankar[1] *, Kossar Pourahmadi[1], Akshayvarun Subramanya[2], and Hamed Pirsiavash[1]

[1] University of California, Davis
[2] University of Maryland, Baltimore County

Here, we provide additional results and analysis on self-supervised (section A), cross modal constraint (section B), semi-supervised (section C) and supervised (section D) settings. Additional results include analysis of retrieved far neighbors (Figs. A1 and A2) and ablations to justify various design choices (tables A5, A6, A7, A8, A10, A12, A13). More details on implementation (section E) and compute calculation (Eq. 1, table A11) are provided. We also provide the code as part of the supplementary materials.

A Results on Self-supervised Setting

A.1 Using far neighbors in unconstrained MSF:

In $\text{CMSF}_{\text{self}}$, we use augmented images from previous epoch to obtain distant neighbors. A trivial way to sample farther neighbors is to just increase the number of neighbors k in the original (unconstrained) MSF [15] method. Here we train MSF with $k = 500$ to compare with our $\text{CMSF}_{\text{self}}$. We train all models for 80 epochs. Settings are similar to our self-supervised settings in section 3.1. For fair comparison, we use memory-bank of size $256k$ for MSF while we use $128K$ for $\text{CMSF}_{\text{self}}$. Results are in Table A.1. While increasing k in MSF helps to sample far NNs, it degrades the accuracy. We hypothesize that this happens due to reducing purity of top-k in unconstrained MSF with increasing k (also shown in Fig. 5 of the main submission). This experiment shows that it is not trivial to to sample far NNs with good purity.

A.2 Effect of number of neighbors k

$\text{CMSF}_{\text{self}}$ uses top-k NNs as part of loss calculation. Here we study the effect of k in $\text{CMSF}_{\text{self}}$ performance. We set $k' = k$ in all models. Note that as described in Line 350 of the main submission, k' is the number of NNs retrieved from the second memory bank M'. We train all models for 80 epochs. All settings are similar to that of $\text{CMSF}_{\text{self}}$ in Section 3.1. Results are shown in Table A.2. Higher values of k and k' degrade model performance. We thus use $k' = k = 5$ in all our main experiments.

we experimented with ResNet-18 and ResNet-101 networks. ResNet-101 is trained for 100 epochs. CMSF outperforms MSF by 1.9 points (51.7% vs 49.8%) with ResNet-18 and 0.8% points (71.9% vs 71.1%) with ResNet-101.

Table A1. Using far neighbors in unconstrained MSF [15]: Using far NNs by trivially increasing k in MSF baseline degrades the accuracy. This is due to the low purity of far NNs in MSF when no constraint is utilized. However, $\text{CMSF}_{\text{self}}$ achieves high accuracy while using distant NNs.

	MSF [15] Top-$k = 500$	MSF [15] Top-$k = 5$	$\text{CMSF}_{\text{self}}$ Top-$k = k' = 5$
NN	35.8	49.7	51.4
20-NN	40.2	54.0	55.5

Table A2. Effect of k in top-k NNs sampling within M: We set $k = k'$ in all models and varied k. We use memory bank of size $256k$. Increasing k degrades the accuracy of the model.

	k'=k=5	k'=k=10	k'=k=20	k'=k=50
NN	51.4	51.3	51.1	49.5
20-NN	55.5	55.3	55.3	53.8

A.3 Results with Different Architectures

In addition to the ResNet-50 architecture, we experiment with a smaller and a larger backbone architecture. We consider ResNet-18 and ResNet-101 networks. The results are shown in table A3. The proposed $\text{CMSF}_{\text{self}}$ improves over MSF across different architectures. Note that the networks are trained only for 100 epochs on ResNet-101.

Table A3. Results with different backbone architectures: We compare performance of our method with that of CMSF with ResNet architectures of different sizes. We observe that CMSF consistently outperforms MSF. * models were trained for 100 epochs instead of 200.

Method	ResNet-18	ResNet-50	ResNet-101*
MSF [15]	49.8	72.2	71.1
$\text{CMSF}_{\text{self}}$	51.7	73.0	71.9

B Cross-modal Constraint

Fig. 6 of main submission showed that proposed CMSF_{sup} is more robust to noisy labels. Here, we explore another such noisy constraint: a pre-trained SSL model from another modality. We consider an unlabeled video dataset and use

the RGB and optical flow inputs as the two different modalities. We first train two SSL models on the RGB and Flow modalities separately using InfoNCE method [21,11]. Then we continue the training on one modality while freezing the other modality and using it as a constraint. In training the flow network using RGB network as constraint, we sample k' nearest neighbors in RGB's memory bank and then search for top-k nearest neighbors among those samples in the memory bank corresponding to Flow.

Implementation Details. Following [11], we use split-1 of UCF-101 [26] (13k videos) as the unlabeled dataset. We use similar augmentation and pre-processing as [11] and calculate optical-flow using unsupervised TV-L1 [30] algorithm. For cross-modal experiments, we use S3D [29] architecture with the input size of 128×128 pixels. We initialize from the pretrained weights of InfoNCE (400-epoch) released by [11]. We use following settings for our method: memory bank of size 8192, $n = 10$, $k = 5$, batch size 128, weight decay $1e-5$, initial lr of 0.001, and learning rate decay by factor of 10 at epoch 80. We train each modality for additional 100 epochs using PyTorch Adam optimizer. For a fair comparison, we run CoCLR using their official code by initializing it from the same model as ours. We use the code from [11] for linear evaluation.

Results: The results are shown in Table A4. We report top-1 accuracy for linear classification and recall@1 for retrieval on the extracted features of frozen networks. All experiments use spatio-temporal 3D data either in RGB or flow format. At the end of 3 stages of training on Flow modality, our method outperforms CoCLR [11] baseline and MSF with 2 stages.

C Results on Semi-supervised Setting

Unless specified, we use the ImageNet100 dataset for all the ablations on the semi-supervised setting for faster experimentation.

C.1 Role of Confidence Threshold in Pseudo-labeling

We use a MLP classification head to predict pseudo-labels for the unlabeled set. As shown in Fig. A1, the accuracy of the classifier is low in the initial stages and improves as training progresses. Using constraints from incorrectly labeled samples might affect the learning process. Thus, we use confidence (class probabilities) based thresholding to select the samples to be used for pseudo-labeling. Only those samples with confidence higher than the threshold are assigned a pseudo-label. Fig. A1 shows that the accuracy of the classifier on the confident samples remains high throughout training, limiting the number of incorrect pseudo-labels. Results for threshold value (t) selection are shown in table A5. As expected, $t = 0$ (i.e, no thresholding) performs poorly compared to higher threshold values. Pseudo-labeling accuracy increases with increasing value of t and the best result is observed for $t = 0.9$. While further increase in

Table A4. Cross-modal constraint: We initialize all models using an InfoNCE pre-trained model. In CMSF-cross modal, one of the modalities is used to constrain and train the other. Superscript indicate the constraint modality, subscript indicate the training modality. For example, in $\text{CMSF}^{\text{Flow}}_{\text{RGB}}$, we continue training CMSF on RGB modality while using frozen pretrained Flow model as the constraint. Note that CoCLR [11] also uses another modality as a constraint in the form of contrastive learning. We continue training InfoNCE SSL model for 200 epochs using MSF [15] for a fair comparison. We use S3D [29] architecture for all models. Models with the final round of training on Flow modality are highlighted with yellow and those on RGB are highlighted with blue. All rows with * contain results for the same model. Results are repeated for easier understanding of the table.

Model	Final modality	Epochs	R@1	Linear
InforNCE_{RGB}	RGB	400	35.5	47.9
$\text{InforNCE}_{RGB} \rightarrow \text{MSF}_{RGB}$	RGB	400+200	39.6	50.8
InforNCE_{Flow}*	Flow	400	45.3	66.1
$\text{InforNCE}_{Flow} \rightarrow \text{MSF}_{Flow}$	Flow	400+200	47.3	64.7
InforNCE_{Flow}*	Flow	400	45.3	66.1
$\text{InforNCE}_{Flow} \rightarrow \text{CoCLR}^{Flow}_{RGB}$	RGB	400+100	49.8	61.0
$\text{InforNCE}_{Flow} \rightarrow \text{CoCLR}^{Flow}_{RGB} \rightarrow \text{CoCLR}^{RGB}_{Flow}$	Flow	400+100+100	50.0	67.3
InforNCE_{Flow}*	Flow	400	45.3	66.1
$\text{InforNCE}_{Flow} \rightarrow \text{CMSF}^{Flow}_{RGB}$	RGB	400+100	45.8	58.1
$\text{InforNCE}_{Flow} \rightarrow \text{CMSF}^{Flow}_{RGB} \rightarrow \text{CMSF}^{RGB}_{Flow}$	Flow	400+100+100	54.1	71.2

t could result in higher pseudo-labeling accuracy, it would also mean that fewer samples are assigned pseudo-labels. Thus, we use $t = 0.9$ in all our experiments on ImageNet100. Since ImageNet-1k has ten times more classes, we reduce the value to 0.85 for all our experiments on ImageNet-1k.

C.2 Effect of Caching on Pseudo-label Training

In addition to optimizing the query encoder network using CMSF loss, we train the pseudo-label classifier head at the end of each epoch of query encoder training. Each round of pseudo-label training entails 40 epochs of classifier head training on the supervised subset of the data (10%). While the time required for backward pass is minimal since only the MLP head is updated, forward pass through the encoder adds significant computational overhead. We thus employ encoder feature caching to overcome this issue. We experiment with two caching settings - offline caching and online caching. In offline caching, encoder features for all the supervised samples are calculated once at the beginning of pseudo-label training and kept fixed for the remaining 39 epochs. In online caching, encoder features for the supervised samples are cached for each mini-batch during the query network training. Similar to offline caching, these features are then fixed and used throughout the 40 epochs of pseudo-labeling network training. Offline technique requires one epoch of forward pass through the encoder, but has the

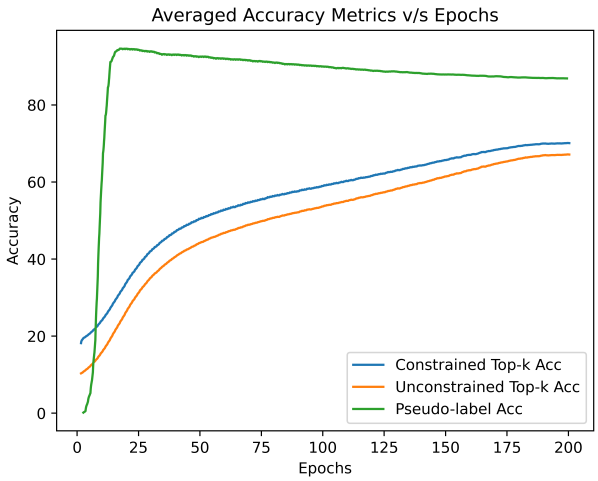

Fig. A1. Unconstrained NN accuracy in semi-supervised: For analysis, we track the pseudo-labeling accuracy and accuracy of top-k neighbors chosen with and w/o applying the pseudo-label based constraint during training. The more accurate constrained NNs provide a better training signal. Pseudo-label accuracy on confident samples remains high throughout training, decreasing slightly as more confident samples are added.

Table A5. Role of confidence threshold in pseudo-labeling (ImageNet100 results): Using confidence based threshold to pseudo-label helps improve performance by eliminating noisy pseudo-labels. A higher threshold value results in higher pseudo-label accuracy but also limits the number of samples that participate in constraint selection. We set the value of t to 0.9 on the ImageNet100 dataset and to 0.85 on the more diverse (1000 classes) ImageNet-1k dataset.

Threshold (t)	1-NN	20-NN	Top-1
0	67.9	71.2	76.2
0.7	67.9	72.3	77.1
0.9	**69.0**	**72.7**	**77.5**

advantage of using the most recent model parameters for feature calculation. Online caching results in features for different images being calculated using different encoder parameters. We observe that both these settings perform similarly on the ImageNet100 dataset (refer table A6). We thus use the online version in all our experiments since it has almost no overhead. With this setting, pseudo-label training increases the training time of each epoch approximately by just 40 seconds.

Table A6. Feature caching for pseudo-label classifier training (ImageNet100 results): We experiment two different caching schemes for pseudo-label training - offline and online. In offline caching, the features are calculated once at the beginning each round of pseudo-label training while in the online setting, the features are cached for each mini-batch during query encoder training. Since both approaches have similar performance, we use the online version since it has minimal computational overhead.

Method	1-NN	20-NN	Top-1
Offline Caching	67.9	71.2	76.2
Online Caching	66.5	71.9	76.0

C.3 Pseudo-label Classifier Selection

The classifier used to generate pseudo-labels plays a crucial role in obtaining effective constraint sets for $\text{CMSF}_{\text{semi}}$. We experiment with two classification techniques - k-NN classifier and MLP classifier trained with cross-entropy loss. Results on ImageNet100 dataset are shown in table A7. k-NN classifier has lower pseudo-labeling accuracy and thus results in poorer performance. We additionally experiment with linear, two and three layer architectures for the MLP classifier head. As shown in table A7, multi-layer head significantly outperform the linear classifier. Since there is minimal difference in performance of two and three layer MLPs, we use a two layer MLP head in all our experiments.

C.4 Fine-tuning without Pseudo-labels

Since we do not explicitly optimizer our encoder networks on the label classification task in the pre-training stage, we perform two-stage fine-tuning. We initially fine-tune the pretrained model on only the supervised samples and use the fine-tuned model to obtain pseudo-labels for the unsupervised ones. The combined data is then used to fine-tune the network again. In table A8, we present results with just a single round of fine-tuning with the 10% supervised samples on ImageNet-1k. Two rounds of fine-tuning provides a small improvement in performance over the single-stage version.

Table A7. Pseudo-label classifier selection (ImageNet100 results): We experiment with different classifier methods and architectures for pseudo-label prediction. Linear layer or multi-layer perceptron (MLP) heads trained using cross-entropy loss on the supervised examples outperform a k-NN classifier. MLP classifiers achieve higher accuracy on the pseudo-labeling task on both the train and test sets. We use a two layer MLP head based classifier in all our experiments.

Pseudo-label Classifier	1-NN	20-NN	Top-1
k-NN Classifier	64.9	69.5	74.7
Linear Classifier	65.6	70.0	75.5
2 Layer MLP Head	67.9	71.2	76.2
3 Layer MLP Head	67.1	71.0	76.1

Table A8. Role of network fine-tuning on classification performance (ImageNet-1k results): We evaluate the trained models using the linear evaluation technique commonly employed for evaluating self-supervised approaches and entire network fine-tuning as performed in semi-supervised methods. Both the methods use 10% of the dataset as supervision. We observe an increase in classification performance when both the encoder and MLP classifier are fine-tuned.

Fine-tune Method	Top-1
Linear layer training	76.5
Full network fine-tune	76.9

D Results on Supervised Setting

D.1 Coarse-grained ImageNet

CMSF_{sup} top-k groups together only top-k neighbors and thus can help in preserving the latent structure of the data compared to top-*all*. To verify this, we consider a dataset with coarse-grained labels where this difference is pronounced. ImageNet dataset was constructed using the WordNet hierarchy. Consider the subtree of WordNet that contains all the 1000 categories from ImageNet-1k as the leaf nodes. To obtain a coarse-grained version, we merge each category in the leaf node to its parent node. After merging, we further ensure that no two of the newly obtained super-classes are in the same path in the graph by merging the descendant into the ancestor class. The total number of classes is thus reduced from 1000 in ImageNet-1k to 93 in our ImageNet-coarse. We train $\text{CMSF}_{\text{self}}$and the baseline approaches in a supervised manner using the coarse labels and then evaluate on the fine-grained (i.e., original) labels on ImageNet-1k validation set. The training settings remain same as that of CMSF_{sup} in Sec.3.2 of main submission.

In Table A9 we compare the top-*all*, top-1000 and top-k variants on the coarse grained version of ImageNet. We consider the top-1000 variant to limit the effect of dataset imbalance introduced due to the merging of classes. CMSF_{sup} top-k sees a minor drop in performance compared to training on ImageNet-1k.

Table A9. Supervised learning on coarse grained ImageNet: We train on the coarse grained version of ImageNet (93 super categories) and perform linear evaluation on the original ImageNet-1k validation set with fine-grained labels (1000 categories). CMSF_{sup} top-10 outperforms all other variants and baselines.

Train Dataset	ImageNet-1k Validation Set				
	Xent	SupCon	CMSF_{sup} top-*all*	CMSF_{sup} top-1000	CMSF_{sup} top-10
ImageNet-1k	77.2	**77.5**	75.7	-	76.4
ImageNet-coarse	61.4	58.7	67.0	71.0	**74.2**

However, methods in which most or all samples in a class are explicitly brought closer - CMSF_{sup} top-*all* and top-1000, cross-entropy and supervised contrastive - see a huge drop in accuracy.

D.2 Ablations

We explore different design choices and parameters of our method and baselines. We add the techniques used for our methods to the baselines to isolate the effect of different losses. The results are reported in Table A10. Training and evaluation details are the same as in Section 3.2 of the main submission.

E Implementation Details

E.1 Transfer Learning

We use the LBFGS optimizer (max_iter=20, and history_size=10) along with the Optuna library [2] in the Ray hyperparameter tuning framework [18]. Each dataset gets a budget of 200 trials to pick the best parameters on validation set. The final accuracy is reported on a held-out test set by training the model on the train+val split using the best hyperparameters. The hyperparameters and their search spaces (in loguniform) are as follows: iterations $\in [0, 10^3]$, lr $\in [10^{-6}, 1]$, and weight decay $\in [10^{-9}, 1]$. We also show that we can reproduce the transfer results for BYOL [10] and SimCLR [6] with our framework. The features are extracted with the following pre-processing for all datasets: resize shorter side to 256, take a center crop of size 224, and normalize with ImageNet statistics. No training time augmentation was used.

E.2 Supervised Setting

Implementation Details of Baselines SupCon: The MLP architecture for SupCon baseline is: linear (2048x2048), batch norm, ReLU, and linear (2048x128). To optimize the SupCon baseline, following [14], we use the first 10 epochs for learning-rate warmup. For both SupCon and ProtoNW, the temperature is 0.1.

Table A10. Ablations of baselines and CMSF$_{\text{sup}}$: All experiments use 200 epochs if not mentioned and use ImageNet-1k dataset. **(a)** More epochs does not improve transfer accuracy for Xent. Thus, the model available from PyTorch [1] (last row) has the best transfer accuracy; **(b)** We add components of our method to improve SupCon baseline. The baseline implementation of SupCon uses std. aug and 16k memory size and it does not include the target embedding u in the positive set. **(c)** We find that our method is not very sensitive to the size of memory bank or top-k in supervised settings; **(d)** Interestingly, excluding the target embedding u from C does not hurts the results. Note that when we do not include the target, the nearest neighbors are still chosen based on the distance to the target, so they will be close to the target.

Method	Mean Trans	Linear IN-1k
(a) Xent		
lr=0.05, cos, epochs=200, strong aug.	71.5	77.2
lr=0.05, cos, epochs=200, std. aug.	71.0	77.3
lr=0.10, cos, epochs=200, strong aug.	72.3	77.1
lr=0.05, cos, epochs=90, std. aug.	72.4	76.8
lr=0.10, cos, epochs=90, std. aug.	74.0	76.7
lr=0.10, step, epochs=90, std. aug.	74.9	76.2
(b) SupCon		
Base SupCon	77.2	77.9
+ change to strong aug.	77.9	77.4
+ add target to positive set	77.8	77.4
+ change to weak/strong aug.	77.8	77.2
+ increase mem size to 128k	78.4	77.5
(c) CMSF$_{sup}$		
top-1 (BYOL-asym)	74.3	69.3
mem=128k, top-2	78.4	76.2
mem=128k, top-10	80.1	76.4
mem=128k, top-20	79.9	76.3
mem=128k, top-*all*	80.1	75.7
mem=512k, top-10	79.9	76.2
mem=512k, top-20	80.1	76.3
(d) CMSF$_{sup}$		
target in top-10	80.1	76.4
target not in top-10	80.3	76.4

Table A11. ResNet50 backbone training FLOPs calculation: We provide the number of forward and backward passes per image (including multi-crops) and the total such passes for the entire training stage. Mean Shift and the proposed constrained mean shift methods have the least compute requirement among all approaches. Eq. 1 provides the formula to calculate the total number of passes and FLOPs. In PAWS, *sup* refers to the size of the support set in the mini-batch.

Method	Unlabeled			Labeled			Mini-Batch	Iters per epoch	Epochs	Total Pass ($\times 10^8$)	FLOPs ($\times 10^{18}$)
	Fwd	Bwd	BS	Fwd	Bwd	BS					
Mean Shift [15]	2	1	256				768	5004	200	7.7	4
BYOL [10]	4	2	4096				24576	312	1000	76.7	40
SwAV [5]	3.1	3.1	4096				25395	312	800	63.4	37
SimCLRv2 [7]	2	2	4096				16384	312	800	40.9	16
UDA† [28]	2	1	15360	1	1	512	47104	40000		18.8	10
FixMatch† [25]	2	1	5120	1	1	1024	17408	250	300	13.1	70
MPL† [23]	3	2	2048	2	2	128	10752	500000		53.8	30
PAWS (sup=6720) [3]	3.1	3.1	4096	1	1	6720	38835	312	300	36.6	21
PAWS (sup=1680) [3]	3.1	3.1	256	1	1	1680	4947	5004	100	24.8	15
PAWS (sup=400) [3]	3.1	3.1	256	1	1	400	2387	5004	100	12.0	7
$\text{CMSF}_{\text{semi}}$-basic	2	1	256				768	5004	200	7.7	4
$\text{CMSF}_{\text{semi}}$	2	1	256				768	5004	200	7.7	4
$\text{CMSF}_{\text{semi}}$-mix prec.	2	1	768				2304	1668	200	7.7	4

Prototypical Networks (ProtoNW): In order to further study the effect of contrast, we design another contrastive version of our top-*all* variation. We calculate a prototype for each class by averaging all its instances in the memory bank. Then, similar to prototypical networks [24], we compare the input with all prototypes by passing their temperature-scaled cosine distance through a SoftMax layer to get probabilities. Finally, we minimize the cross-entropy loss. Note that this method is still contrastive in nature because of the SoftMax operation.

E.3 Semi-supervised Setting

Pretraining: Similar to the self-supervised setting, we train the network for 200 epochs using SGD optimizer (batch size=256, lr=0.05, momentum=0.9, weight decay=1e-4). Ten nearest neighbors are chosen from the constraint set for loss calculation. The size of memory bank is set to 128000. We train the pseudo-label classifier using an additional SGD optimizer (batch size=256, lr=0.01, momentum=0.9, weight decay=1e-4) for 10 epochs at the end of each epoch of query encoder training. A confidence threshold value of 0.85 is used to assign pseudo-labels to the unlabeled samples.

Fine-tuning: In addition to pretraining, we use a two layer MLP atop the CNN backbone and fine-tune the entire network on the supervised subset for 20 epochs. This fine-tuned network is used to pseudo-label the unlabeled set with a confidence threshold of 0.9. Samples above the threshold are combined with

Table A12. Noisy supervised setting on ImageNet-100: Our method is more robust to noisy annotation compared to Xent and SupCon. The top-*all* variant suffers greater degradation compared to top-10 since all images from a single category are not guaranteed to be semantically related in the noisy setting.

Method	Noise	Food 101	CIFAR 10	CIFAR 100	SUN 397	Cars 196	Air- craft	DTD	Pets	Calt. 101	Flwr 102	**Mean Trans**	**Linear IN-100**
Xent	0%	53.6	81.9	61.1	37.8	25.7	29.5	56.9	69.7	70.2	82.3	56.9	85.7
SupCon	0%	61.5	88.7	69.0	49.1	51.6	48.2	65.4	81.0	87.0	89.8	69.1	86.9
$CMSF_{sup}$ top-*all*	0%	61.6	88.2	68.5	49.9	54.6	52.7	64.7	82.2	89.6	89.1	70.1	84.9
$CMSF_{sup}$ top-10	0%	62.6	86.8	66.2	50.5	54.7	51.0	64.6	82.4	88.5	90.4	69.8	85.0
Xent	5%	46.5	81.1	58.1	35.8	27.5	36.0	58.7	67.5	73.3	77.0	56.1	81.5
SupCon	5%	60.0	87.1	66.4	48.2	52.1	47.8	65.1	80.8	85.7	89.3	68.3	85.7
$CMSF_{sup}$ top-*all*	5%	60.3	87.5	66.4	49.1	55.5	53.0	64.8	80.9	87.3	89.9	69.5	84.4
$CMSF_{sup}$ top-10	5%	61.6	86.8	67.4	49.6	55.8	51.2	63.4	81.5	86.7	90.6	69.5	84.7
Xent	10%	44.1	79.5	56.1	32.4	26.1	34.5	56.1	69.7	72.5	75.1	54.6	79.6
SupCon	10%	58.8	85.8	66.4	47.0	50.6	47.7	65.3	79.8	85.0	89.1	67.6	84.0
$CMSF_{sup}$ top-*all*	10%	59.4	86.4	66.0	48.8	55.0	51.4	64.7	80.1	87.8	89.0	68.9	83.1
$CMSF_{sup}$ top-10	10%	60.9	87.2	66.9	49.4	54.2	51.4	65.5	80.6	88.5	90.0	69.5	83.8
Xent	25%	49.0	77.2	54.5	30.6	25.9	30.7	53.1	66.6	64.1	77.8	53.0	75.2
SupCon	25%	55.6	84.9	63.4	43.1	43.9	43.7	62.9	74.3	82.1	86.8	64.1	81.1
$CMSF_{sup}$ top-*all*	25%	56.4	85.7	64.2	46.0	53.6	49.6	62.7	74.2	85.2	87.4	66.5	78.8
$CMSF_{sup}$ top-10	25%	58.9	85.2	64.9	47.8	55.0	50.6	64.0	80.0	86.3	89.7	68.2	81.8
Xent	50%	44.4	72.3	51.3	31.1	21.4	24.9	46.0	57.4	56.0	73.0	47.8	67.8
SupCon	50%	30.8	64.9	38.9	24.2	13.6	20.5	45.5	55.2	60.1	59.2	41.3	69.0
$CMSF_{sup}$ top-*all*	50%	44.7	79.3	54.9	35.2	35.7	41.2	54.9	54.6	75.3	75.1	55.1	61.6
$CMSF_{sup}$ top-10	50%	58.7	85.7	64.2	47.5	51.6	50.5	62.0	77.3	86.8	70.1	65.4	80.1

Table A13. Transfer dataset details: Train, val, and test splits of the transfer datasets are listed in this table. **Test split:** We follow the details in [15]. For Aircraft, DTD, and Flowers datasets, we use the provided test sets. For Sun397, Cars, CIFAR-10, CIFAR-100, Food101, and Pets datasets, we use the provided val set as the hold-out test set. For Caltech-101, 30 random images per category are used as the hold-out test set. **Val split:** For DTD and Flowers, we use the provided val sets. For other datasets, the val set is randomly sampled from the train set. For transfer setup, to be close to BYOL [10], the following val set splitting strategies have been used for each dataset: Aircraft: 20% samples per class. Caltech-101: 5 samples per class. Cars: 20% samples per class. CIFAR-100: 50 samples per class. CIFAR-10: 50 samples per class. Food101: 75 samples per class. Pets: 20 samples per class. Sun397: 10 samples per class. **Accuracy measure:** *Top-1* refers to top-1 accuracy while *Mean* refers to mean per-class accuracy.

Dataset	Classes	Train samples	Val samples	Test samples	Accuracy measure	Test set provided
Food101 [4]	101	68175	7575	25250	Top-1	-
CIFAR-10 [17]	10	49500	500	10000	Top-1	-
CIFAR-100 [17]	100	45000	5000	10000	Top-1	-
Sun397 (split 1) [27]	397	15880	3970	19850	Top-1	-
Cars [16]	196	6509	1635	8041	Top-1	-
Aircraft [19]	100	5367	1300	3333	Mean	Yes
DTD (split 1) [8]	47	1880	1880	1880	Top-1	Yes
Pets [22]	37	2940	740	3669	Mean	-
Caltech-101 [9]	101	2550	510	6084	Mean	-
Flowers [20]	102	1020	1020	6149	Mean	Yes

the supervised set for a second round of fine-tuning for 20 epochs. We observe that nearly one third of the samples in the dataset have confidence higher than the threshold at the end of the first fine-tuning stage. We use a SGD optimizer (batch size=256, lr=0.005, momentum=0.9, weight decay=1e-4) for both the fine-tuning stages. The learning rate is multiplied by 0.1 at the end of epoch 15.

Calculation of forward and backward FLOPs: In figure 1 of the main submission, we present a plot of top-1 accuracy against total compute and resources for various semi-supervised approaches. Here (table A11) we present the calculation of the forward and backward FLOPS for each of the methods. We set the backward FLOPs to be twice the forward number of FLOPs [12] for a single image and the total FLOPs to be the sum of forward and backward pass FLOPs for the entire training. We use a value of 3.9 GFLOPs for a single forward pass of 224×224 resolution image through the ResNet50 backbone [13]. Additional compute due to the use of multi-crops are accounted for. A scalar multiplier of $(\frac{K}{224})^2$ is used for images of resolution $K \times K$ (e.g., using one (96×96) image would be equivalent to 0.184 image of resolution (224×224)).

However, we do not consider the floating point precision (mixed or full precision) in our calculations. We show that similar performance can be achieved by using both automatic mixed precision and full precision floating point during training (table 4, main submission) and thus focus the compute calculation on the total number of forward and backward passes. Eq. 1 provides the formula to calculate the total number of training passes and FLOPs.

$$
\begin{aligned}
\text{Fwd mini-batch} &= (\text{Unlabeled fwd crops} * \text{Unlabeled batch-size}) \\
&\quad + (\text{Labeled fwd crops} * \text{Labeled batch-size}) \\
\text{Bwd mini-batch} &= (\text{Unlabeled bwd crops} * \text{Unlabeled batch-size}) \\
&\quad + (\text{Labeled bwd crops} * \text{Labeled batch-size}) \\
\text{Fwd passes} &= \text{Fwd mini-batch} * \text{Iterations per epoch} * \text{Epochs} \\
\text{Bwd passes} &= \text{Bwd mini-batch} * \text{Iterations per epoch} * \text{Epochs} \\
\text{Total FLOPs} &= (\text{Fwd passes} + 2 * \text{Bwd passes}) * (3.9 \times 10^9)
\end{aligned}
\tag{1}
$$

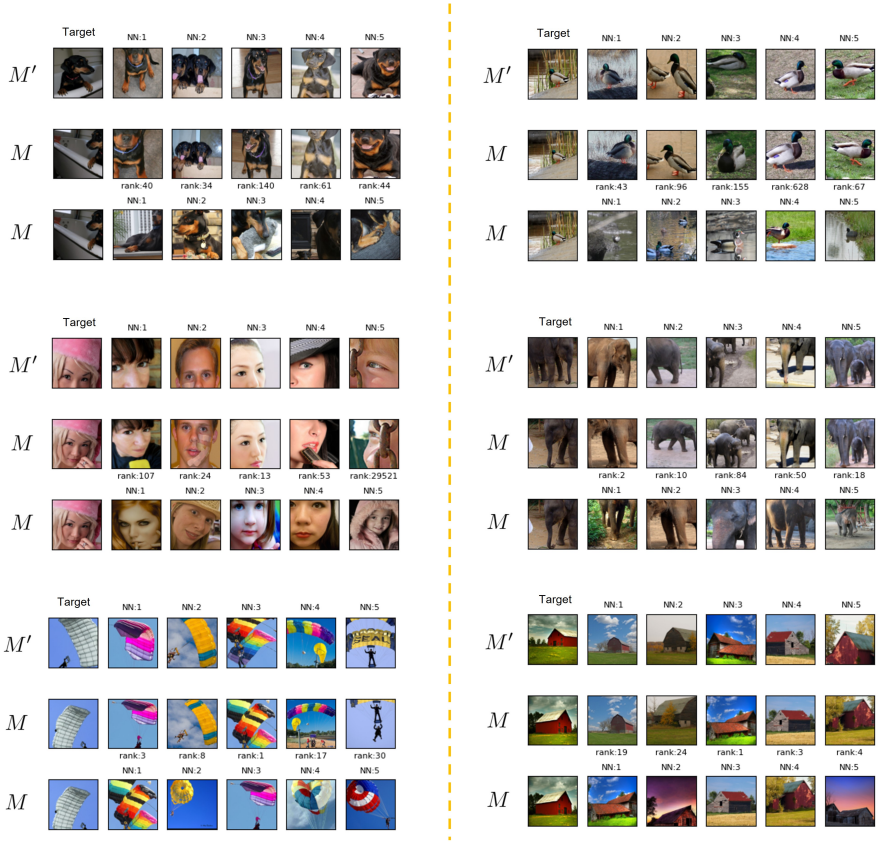

Fig. A2. CMSF$_{\text{self}}$ nearest neighbor selection: We use epoch 100 of CMSF$_{\text{self}}$ to visualize Top-5 NN from primary (M) and auxiliary (M') memory banks. M stores features for the current epoch while M' contains representations from a different augmentation of the same image instance from the previous epoch. First row shows the target image and its top-5 NNs from the auxiliary memory bank M'. Samples of the second row are the images in M corresponding to the ones in row 1. Thus, rows 1 and 2 contain different augmentations of the same image instances. We also report their rank in M in row 2. The last row contains the top-5 NNs in M. Note that constrained samples in M (second row), have high rank while they are semantically similar to the target.

References

1. Torchvision models. https://pytorch.org/docs/stable/torchvision/models.html 9
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 2623–2631 (2019) 8
3. Assran, M., Caron, M., Misra, I., Bojanowski, P., Joulin, A., Ballas, N., Rabbat, M.: Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. ICCV (2021) 10
4. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: European Conference on Computer Vision (2014) 12
5. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: Advances in Neural Information Processing Systems. pp. 9912–9924. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf 10
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020) 8
7. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.E.: Big self-supervised models are strong semi-supervised learners. Advances in Neural Information Processing Systems **33**, 22243–22255 (2020) 10
8. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: Computer Vision and Pattern Recognition (2014) 12
9. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Pattern Recognition Workshop (2004) 12
10. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., et al.: Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733 (2020) 8, 10, 12
11. Han, T., Xie, W., Zisserman, A.: Self-supervised co-training for video representation learning (2021) 3, 4
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 12
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018) 12
14. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. Advances in Neural Information Processing Systems **33** (2020) 8
15. Koohpayegani, S.A., Tejankar, A., Pirsiavash, H.: Mean shift for self-supervised learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10326–10335 (October 2021) 1, 2, 4, 10, 12
16. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D object representations for fine-grained categorization. In: Workshop on 3D Representation and Recognition. Sydney, Australia (2013) 12

17. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009) 12
18. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118 (2018) 8
19. Maji, S., Rahtu, E., Kannala, J., Blaschko, M.B., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013) 12
20. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: Indian Conference on Computer Vision, Graphics and Image Processing (2008) 12
21. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding (2018) 3
22. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and dogs. In: Computer Vision and Pattern Recognition (2012) 12
23. Pham, H., Dai, Z., Xie, Q., Le, Q.V.: Meta pseudo labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11557–11568 (2021) 10
24. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. arXiv preprint arXiv:1703.05175 (2017) 10
25. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. Advances in Neural Information Processing Systems **33** (2020) 10
26. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild (2012) 3
27. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: Computer Vision and Pattern Recognition (2010) 12
28. Xie, Q., Dai, Z., Hovy, E., Luong, M.T., Le, Q.V.: Unsupervised data augmentation for consistency training. NeurIPS (2020) 10
29. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification (2018) 3, 4
30. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime tv-l1 optical flow. In: DAGM-Symposium (2007) 3