Constrained Mean Shift Using Distant Yet Related Neighbors for Representation Learning

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

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

Abstract

We are interested in representation learning in self-supervised, supervised, or semi-supervised settings. The prior work on applying mean-shift idea for self-supervised learning, MSF, generalizes the BYOL idea by pulling a query image to not only be closer to its other augmentation, but also to the nearest neighbors (NNs) of its other augmentation. We believe the learning can benefit from choosing far away neighbors that are still semantically related to the query. Hence, we propose to generalize MSF algorithm by constraining the search space for nearest neighbors. We show that our method outperforms MSF in SSL setting when the constraint utilizes a different augmentation of an image, and outperforms PAWS in semi-supervised setting with less training resources when the constraint ensures the NNs have the same pseudo-label as the query. Our code is available here: https://github.com/UCDvision/CMSF.

1. Introduction

Recently, we have seen great progress in self-supervised learning (SSL) methods that learn rich representations from unlabeled data. Such methods are important since they do not rely on manual annotation of data, which can be costly, biased, or ambiguous. Hence, SSL representations may perform better than supervised ones in transferring to downstream visual recognition tasks.

Most recent SSL methods, *e.g.*, MoCo [33] and BYOL [32], encourage a query image to be closer to its own augmentation compared to some other random images. Follow-up works have focused on improving the positive pairs through generating better augmentations [46, 58, 69] and the negative set by increasing the set size [33] or mining effective samples [38, 40, 75], but have largely ignored possibility of utilizing additional positive images. More recently, [7, 25, 42] expand the positive set using nearest neighbors. Inspired by classic mean-shift algorithm, MSF [42], generalizes BYOL to group similar images together.

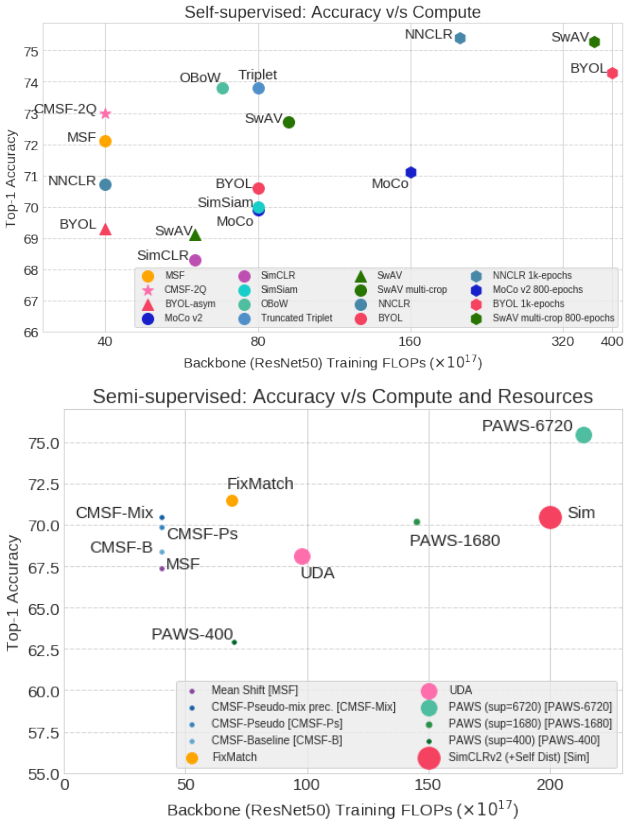

Figure 1. **Accuracy and training compute comparison:** We report the total training FLOPs for forward and backward passes through the CNN backbone. **(Top) Self-supervised:** All methods are trained on ResNet-50 backbone for 200 epochs. CMSF achieves competitive accuracy with considerably lower compute. **(Bottom) Semi-supervised:** Circle radius is proportional to the number of GPUs/TPUs used. In addition to being compute efficient, CMSF is trained with an order of magnitude lower resources, making it more practical and accessible.

*Equal contribution

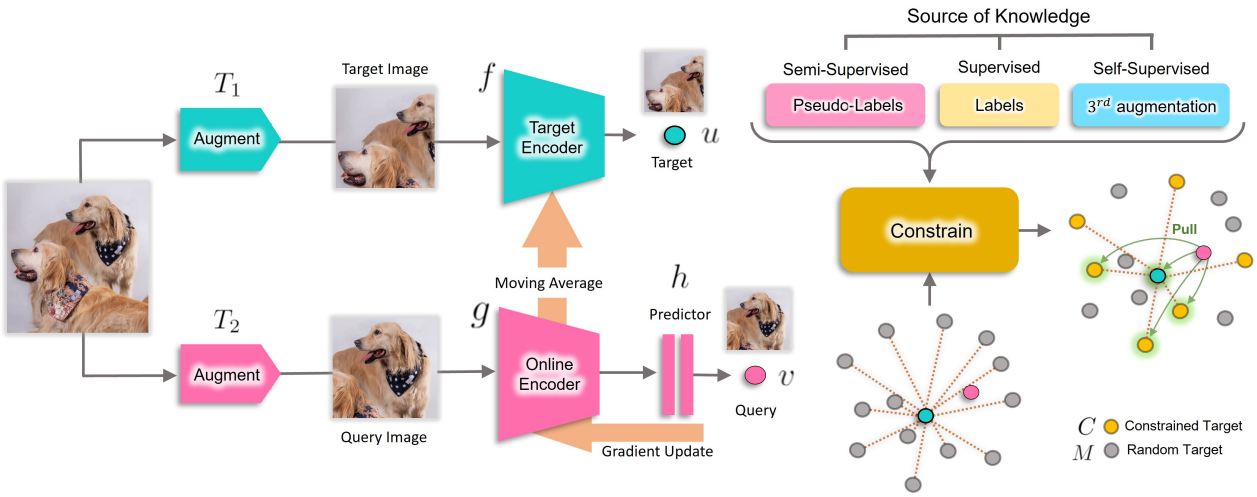

Figure 2. **Our method (CMSF):** We augment an image twice and pass them through online and target encoders followed by ℓ_2 normalization to get u and v. Mean-shift [42] encourages v to be close to both u and its nearest neighbors (NN). Here, we constrain the NN pool based on additional knowledge in the form of supervised labels, classifier or previous augmentation based pseudo-labels. These constraints ensure that the query is pulled towards semantically related NNs that are farther away from the target feature.

MSF pulls a query image to be close to not only its augmentation, but also the top-k nearest neighbors (NNs) of its augmentation.

We argue that the top-k neighbors are close to the query image by construction, and thus may not provide a strong supervision signal. We are interested in choosing far away (non-top) neighbors that are still semantically related to the query image. This cannot be trivially achieved by increasing the number of NNs since the purity of retrieved neighbors decreases with increasing k (refer Fig. 3), where the purity is defined as the percentage of the NNs belonging to the same semantic category as the query image.

We generalize MSF [42] method by simply limiting the NN search to a smaller subset that we believe is semantically related to query. We define this constraint to be the NNs of another query augmentation in SSL setting and images sharing the same label or pseudo-label in supervised and semi-supervised settings.

While we aim to obtain distant samples of the same category, note that we group only a few neighbors (k in our method) from the constrained subset instead of grouping the whole subset together. This is in contrast to cross-entropy supervised learning, where we force all images of a category to form a cluster or be on the same side of a hyper-plane. Our method can benefit from this relaxation by preserving the latent structure of the categories and also being robust to noisy labels.

Our experiments show that the method outperforms the various baselines in all three settings with same or less amount of computation in training. It outperforms MSF [42] in SSL, cross-entropy in supervised (with clean or noisy labels), and PAWS [6] in semi-supervised settings.

2. Method

Similar to MSF [42], given a query image, we are interested in pulling its embedding closer to the mean of the embeddings of its nearest neighbors (NNs). However, since top NNs are close to the target itself, they may not provide a strong supervision signal. On the other hand, far away (non-top) NNs may not be semantically similar to the target image. Hence, we constrain the NN search space to include mostly far away points with high purity. The purity is defined as the percentage of the NNs being from the same semantic category as the query image. We use different constraint selection techniques to analyze our method in supervised, self- and semi-supervised settings.

Following MSF and BYOL, we assume two embedding networks: a target encoder $f(.)$ with parameters θ_f and an online encoder $g(.)$ with parameters θ_g. The online encoder is directly updated using backpropagation while the target encoder is updated as a slowly moving average of the online encoder: $\theta_f \leftarrow m\theta_f + (1-m)\theta_g$ where m is close to 1. This is the momentum idea introduced in [33]. We add a predictor head $h(.)$ [32] to the end of the online encoder so that pulling the embeddings together encourages one embedding to be predictable by the other one and not necessarily encouraging the two embeddings to be equal. In the experiments, we use a two-layer MLP for $h(.)$.

Given a query image x, we augment it twice with $T_1(.)$ and $T_2(.)$, feed them to the encoders, and normalize them with their ℓ_2 norm to get $u = \frac{f(T_1(x))}{||f(T_1(x))||_2}$ and $v = \frac{h(g(T2(x)))}{||h(g(T2(x)))||_2}$. We add u to the memory bank M and remove the oldest entries to maintain a limited size for

Figure 3. **Nearest neighbor selection on constrained memory bank:** We use intermediate checkpoint (epoch 100) of our self-supervised CMSF-KM method to visualize constrained samples during training. First row shows top-5 NNs of target in constrained set C and their corresponding rank in the unconstrained memory bank M. While they are not the closest samples to the target (higher rank index), they are semantically similar to the target. This shows that the constraint can capture far away samples with similar semantic as the target. The second row depicts images from memory bank with one rank lower compared to the corresponding image in the first row. These images contain incorrect category retrievals. Distant neighbors cannot be trivially obtained by increasing the number of NNs. This highlights the importance and effectiveness of proposed constraint selection methods. Examples are chosen randomly without cherry-picking.

M. We define C as a smaller subset of M using the constraint that depends on the setting and will be discussed below. Then, we find top-k neighbors of u in C including u itself and call it $S = \{z_i\}_{i=1}^{k}$. Finally, we update $g(.)$ by minimizing the following loss and update $f(.)$ with the momentum update. Note that self-supervised mean-shift algorithm in [42] is a specific case of our algorithm in which there is no constraint to limit the nearest neighbor search, *i.e.*, $C = M$.

$$L = \frac{1}{k} \sum_{i=1}^{k} v^T z_i$$

In top-*all* variation of our method, k is equal to the total size of C. Note that since u itself is included in the nearest neighbor search, by limiting the size of the constrained set C to one (top-1), the method will be identical to BYOL [32] and by setting $C = M$, it will be identical to self-supervised mean-shift [42]. Hence, our method covers a larger spectrum by defining the constrained set.

2.1. Self-supervised Setting

In the initial stages of learning two diverse augmentations of an image are not very close to each other in the embedding space. Thus, one way of choosing far away NNs for the target u with high purity is to limit the neighbor search space based on the NNs of a different augmentation u' of the target.

CMSF-KM: Here, we perform clustering at the end of each epoch (using the cached embeddings of that epoch) and define C to be a subset of M that shares the same cluster assignment as the target. Similar to MSF, we then use top-k NNs of target u from constrained set C for loss calculation to maintain high purity. Since augmentations are chosen randomly and independently at each epoch, cluster assignment and distance minimization happen with different augmentations. Even though members of a cluster are close to each other in the previous epoch, the set C may not be close to the current target. This improves learning by averaging distant samples with a good purity.

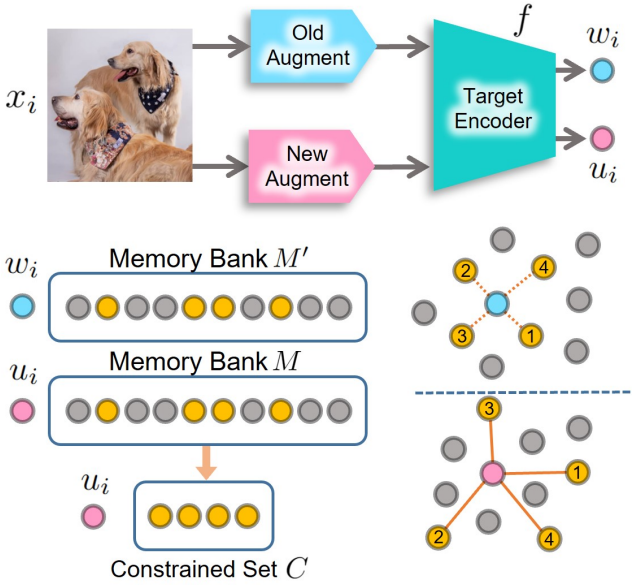

Figure 4. **CMSF-2Q:** The NNs of previous augmentation are obtained in the memory bank M' and corresponding indices in current memory bank M are used to constrain MSF.

CMSF-2Q: We propose this method (refer Fig. 4) to show the importance of using a different augmentation to constrain the NN search space. In addition to M, we maintain a second memory bank M' that is exactly the same as M but containing a different (3^{rd}) augmentation of the query image. We assume $w_i \in M'$ and $u_i \in M$ are two embeddings corresponding to the same image x_i. Then, for image x_i, we find NNs of w_i in M' and use their indices to construct the search space C from M. Note that the elements in C are not necessarily close to each other since they are chosen to be close to each other under a different augmentation M'. As a result, C will maintain good purity while being diverse (refer Table 1-Right and Fig. 5).

Since it is expensive to embed a third augmentation of each image, we embed only two augmentations as in

MSF and cache the embeddings from the previous epoch, keeping the most recent embedding for each image. Since cache size is equal to dataset size, we store it in CPU memory and maintain the auxiliary memory bank M' by loading the corresponding part of it to the GPU memory for each minibatch. As the target model evolves slowly, the NN search within the cached features (from the previous epoch) will still be valid. Table 1-Right shows that in the intermediate stages of learning, the top elements of C are spread apart in M, and get closer to the top elements of M as the learning progresses.

2.2. Supervised Setting

While the supervised setting is not our primary novelty or motivation, we study it to provide more insights into our constrained mean-shift framework. Since we do have access to the labels of each image, we can simply construct C as the subset of M that shares the same label as the target. This guarantees 100% purity for the NNs.

Note that most supervised methods, including cross-entropy loss, try to group all examples of a category together on the same side of a hyper-plane while remaining categories are on the other side. However, our method pulls the target to be close to only those examples of the same category that are already close to the target. This results in a supervised algorithm that may keep the latent structure of each category which may be useful for downstream tasks, *e.g.*, fine-grained classification.

Moreover, as shown in the experiments (Fig. 6), our method is more robust to label noise since most mis-labeled images will be far from the target embedding, and thus ignored in learning. This motivates applying our method to semi-supervised setting where the limited supervision provides noisy labels.

2.3. Semi-supervised Setting

In this setting, we assume access to a small labeled and a large unlabeled dataset. We train a simple classifier using the current embeddings of the labeled data and use the classifier to pseudo-label the unlabeled data. Then, similar to the supervised setting, we construct C to be the elements of M that share the pseudo-label with the target. Again, this method increases the diversity of C while maintaining high purity. To keep the purity high, we enforce the constraint only when the pseudo-label is very confident (the probability is above a threshold.) For the samples with non-confident pseudo-label, we relax the constraint resulting in regular MSF loss (*i.e.,* $C = M$.) Moreover to reduce the computational overhead of pseudo-labeling, we cache the embeddings throughout the epoch and train a 2-layer MLP classifier using the frozen features in the middle and end of each epoch. Note that training the linear classifier for pseudo-labeling is still contrastive.

3. Experiments

We use PyTorch for all our experiments.

Implementation details: Unless specified, we use the same hyper-parameter values in self-, semi- and fully supervised settings. All models are trained on ImageNet-1k (IN-1k) for 200 epochs with ResNet-50 [34] backbone and SGD optimizer (lr=0.05, batch size=256, momentum=0.9, and weight decay=1e-4) with cosine scheduling of learning rate. The value of momentum for the moving average key encoder in 0.99 for CMSF. The MLP architecture for CMSF is as follows: (linear (2048x4096), batch norm, ReLU, linear (4096x512)). The default memory bank size is 128k. Top-k is set to 10 in the semi- and supervised settings and 5 in the self-supervised setting. To train our self-supervised models, we add CMSF loss to MSF loss. Additional details are provided in the appendix. Our main CMSF experiment with 200 epochs takes nearly 6 days on four NVIDIA-2080TI GPUs.

Evaluation: In SSL and supervised settings, we evaluate the pre-trained models using linear evaluation (*Linear IN-1k*) in both ImageNet classification and transfer settings. The model backbone parameters are fixed and a single linear layer is trained atop them following the setting in CompRess [2]. Additionally, we report k-nearest neighbor ($k = 1, 20$) for the SSL setting as in [2]. The transfer performance is evaluated on the following datasets: Food101 [12], SUN397 [81], CIFAR10 [44], CIFAR100 [44], Cars196 [43], Aircraft [50], Flowers (Flwrs102) [53], Pets [56], Caltech-101 (Calt101) [26], and DTD [22] (additional details in appendix).

3.1. Self-Supervised Learning

CMSF with K-Means (CMSF-KM): We maintain a memory bank equal in size to the dataset (1.2M for ImageNet) to store the features on which k-means clustering is performed. The memory bank features are used for clustering once at the end of each epoch and are thus stored in the CPU. The number of clusters K is set to 50,000.

CMSF with 2Q (CMSF-2Q): Similar to CMSF-KM, we maintain cached embeddings in the CPU with features of each sample in the dataset with the previous augmentation. However, here we employ an additional memory bank M' of the same size as regular memory bank M. The cached features corresponding to the current mini-batch are retrieved from CPU to maintain memory bank M' with previous augmentations. Top-5 NNs of target with old augmentation are obtained from M' and corresponding indices in M are used as constraint for the current target. The features removed from queue M at every iteration are used to update the cached features in the CPU.

Method	Ref.	Batch Size	Epochs	Sym. Loss 2x FLOPS	Multi-Crop Training	Top-1 Linear	NN	20-NN
Supervised	[1]	256	100	-	-	76.2	71.4	74.8
Random-init	-	-	-	-	-	5.1	1.5	2.0
SeLa-v2 [84]	[15]	4096	400	✓	✗	67.2	-	-
SimCLR [17]	[17]	4096	1000	✓	✗	69.3	-	-
SwAV [15]	[15]	4096	400	✓	✗	70.1	-	-
DeepCluster-v2 [14]	[15]	4096	400	✓	✗	70.2	-	-
SimSiam [20]	[20]	256	400	✓	✗	70.8	-	-
MoCo v2 [33]	[19]	256	800	✗	✗	71.1	57.3	61.0
CompRess [2]	[2]	256	1K+130	✗	✗	71.9	63.3	66.8
InvP	[74]	256	800	✗	✗	71.3	-	-
BYOL [32]	[32]	4096	1000	✓	✗	74.3	62.8	66.9
SwAV [15]	[15]	4096	800	✓	✓	75.3	-	-
NNCLR	[25]	4096	1000	✗	✗	**75.4**	-	-
SimCLR [17]	[20]	4096	200	✓	✗	68.3	-	-
SwAV [15]	[20]	4096	200	✓	✗	69.1	-	-
MoCo v2 [33]	[20]	256	200	✓	✗	69.9	-	-
SimSiam [20]	[20]	256	200	✓	✗	70.0	-	-
NNCLR [25]	[25]	4096	200	✗	✗	70.7	-	-
BYOL [32]	[20]	4096	200	✓	✗	70.6	-	-
SwAV [15]	[20]	256	200	✓	✓	72.7	-	-
Truncated Triplet [75]	[75]	832	200	✓	✗	**73.8**	-	-
OBoW [29]	[29]	256	200	✗	✓	**73.8**	-	-
MoCo v2 [33]	[19]	256	200	✗	✗	67.5	50.9	54.3
CO2 [77]	[77]	256	200	✗	✗	68.0	-	-
BYOL-asym [32]	[42]	256	200	✗	✗	69.3	55.0	59.2
ISD [67]	[67]	256	200	✗	✗	69.8	59.2	62.0
MSF (1M) [42]	[42]	256	200	✗	✗	72.4	62.0	64.9
MSF (256K) [42]	[42]	256	200	✗	✗	72.2	62.1	65.1
CMSF-KM (128K)	-	256	200	✗	✗	**72.9**	**63.2**	**66.2**
CMSF-2Q (128K)	-	256	200	✗	✗	**73.0**	**63.2**	**66.4**

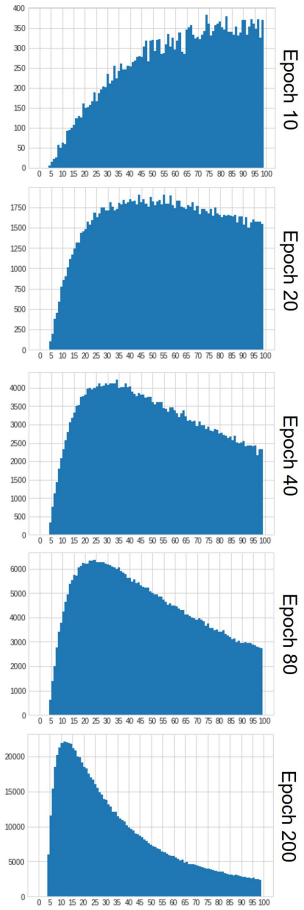

Table 1. **Left: Evaluation on full ImageNet:** We compare our model with other SOTA methods in Linear (Top-1 Linear) and Nearest Neighbor (NN,20-NN) evaluation. We use a 128K memory bank for CMSF and provide comparison with both 256K and 1M memory bank versions of MSF. Since CMSF-2Q uses NNs from two memory banks, it is comparable to MSF (256K) in memory and computation overhead. Our method outperforms other SOTA methods with similar compute including MSF. "Multi-Crop" refers to use of more than 2 augmentations per image during training (*e.g.*, OBoW uses $2 \times 160 + 5 \times 96$ resolution images in both forward and backward passes compared to a single 224 in CMSF). Use of multi-crops significantly increases compute while symmetric loss doubles the computation per batch. Thus methods employing these strategies are not directly comparable with CMSF. We additionally train a combined model with both K-Means and 2Q as the constraint achieving 73.1% Linear, comparable to both CMSF-KM and CMSF-2Q. This suggests that both approaches gain similar information by utilizing previous augmentations for designing the constraint. **Right: Histogram of constrained sample ranks:** We consider the 5^{th} NN in the constrained set C and obtain its rank in the unconstrained memory bank M. The histogram of these ranks are shown upto rank 100 for different train stages of CMSF-2Q. A large number of distant neighbors are part of constraint in the early stages of training while there is a higher overlap between constrained and unconstrained NN set towards the end of training.

3.1.1 Results

Results on ImageNet: Results for CMSF-KM and CMSF-2Q are shown in table 1. Both variants outperform MSF baseline with a larger memory bank. This empirically supports our idea of bringing together far yet semantically similar samples (Fig 3) by constraining the neighbor search space on the memory bank. CMSF methods also achieve state-of-the-art performance on both NN and linear metrics when compared to approaches with similar computational budget. We compare our method to other state-of-the-art approaches with 200 epochs of training in Fig. 1 and observe a good trade-off in terms of accuracy and compute for the proposed CMSF method. Note that we train without symmetrical loss and multi-crop strategy which are known to generally improve performance [15] at the cost of significant increase in compute.

Evaluation on ImageNet Subsets: Following [17, 35], we evaluate the pre-trained models on the ImageNet classification task with limited labels. We report results

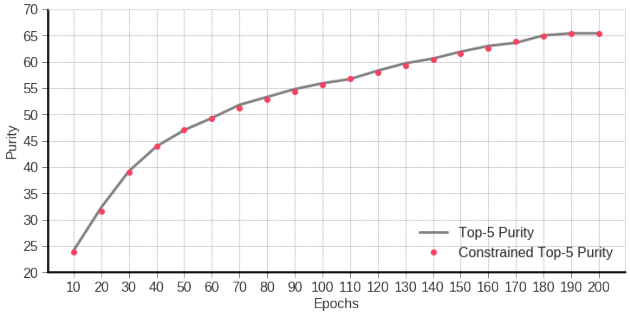

Figure 5. **Purity of constrained samples:** We plot purity of Top-5 samples in constrained memory bank C and Top-5 samples in unconstrained memory bank M for CMSF-2Q during training. Both sets have similar purity values at all stages of training.

with 1% and 10% labeled subsets of ImageNet (table 3). CMSF-2Q and CMSF-KM outperform MSF and are comparable to existing approaches requiring significantly higher training time.

Transfer Learning: We follow the procedure in [17, 32] for transfer evaluation (refer table 2). Hyperparameters for each dataset are tuned independently based on the validation set accuracy and final accuracy is reported on the held-out test set (more details in suppl.). CMSF transfers well to all datasets with CMSF-KM achieving state-of-the-art average performance among methods trained for 200 epochs.

Purity of constrained samples: In both the CMSF-2Q and CMSF-KM methods, we depend on information from previous augmentations to constrain NN search in the current memory bank. Our goal is to improve learning by using distant samples with a good purity. We observe that the top-k samples from constrained memory bank C have high rank in M (are far neighbors of the target)(Table 1-Right), and that they have similar purity as the top-k samples from unconstrained memory bank M (Fig. 5) during training. As a result, C maintains good purity while being diverse.

Effect of k in Top-k for constrained memory bank: Since the constraint is noisy in the SSL setting, it is important to select top NNs from C. We study this by using all samples of the C in our CMSF-KM model. We observe that the 20-NN accuracy decrease from 66.2% to 62.9% when using all samples in C instead of only top 5 NNs. This is aligned with our observation in noisy supervised settings in Fig 6.

3.2. Supervised Learning

3.2.1 Baselines

Cross entropy (Xent): Xent [11, 48, 59] is a popular method for training standard supervised models.

Supervised Contrastive (SupCon): SupCon extends the instance discrimination framework from SSL to supervised learning [41]. It is a contrastive setting in which the positive set contains all images from the same category. The top-all variation of CMSF is similar to SupCon without contrast.

Prototypical Networks (ProtoNW): We design a baseline similar to prototypical networks [62] where class prototypes are obtained by averaging the features of all instances from the class in the memory bank (more details in appendix).

Evaluation: Unlike Xent baseline, SupCon, ProtoNW, and CMSF do not train a linear classifier during the pre-training stage. Thus we use the pre-training dataset ImageNet-1k (IN-1k) for linear evaluation of the frozen features as done in SSL. For Xent, we use the linear classifier trained during pre-training. We use the same settings and datasets as self-supervised for transfer learning evaluation.

Results: Results on IN-1k dataset are shown in table 2. In top-all variation of our method, k is equal to the total size of C. SSL inspired methods like CMSF and SupCon significantly outperform Xent when trained for similar number of epochs. We observe that improvements in ImageNet performance do not always translate to transfer performance. Interestingly, CMSF performs the best on transfer evaluation, particularly on fine-grained datasets like Cars196 and Aircraft. We believe that the absence of explicit cross-entropy based optimization using the supervised labels preserves the multi-modal distribution of categories improving fine-grained performance. Supervised CMSF uses class labels only as a constraint for MSF during pre-training and does not explicitly optimize on the classification task. Superior performance of CMSF-top-10 demonstrates the importance of using distant yet semantically related neighbors as positives.

Noisy Labels: In the noisy setting, labels of a fixed percentage of images are randomly corrupted and kept constant throughout training. We consider, 5%, 10%, 25% and 50% label corruption (noise) rates. For faster experiments, we report results on the ImageNet-100 dataset [68] (Fig. 6). We observe a significantly higher degradation in performance of Xent baseline and CMSF-top-all compared to CMSF-top-10 at high noise levels. The gap between the approaches is larger on transfer learning. These observations indicate that nearest neighbor based methods like CMSF are better suited for noisy constraint settings compared to approaches utilizing all samples of a class as positives. This robustness to label noise motivates our application of CMSF to self- and semi-supervised settings. Purity of top-k NNs of previous augmentation and pseudo-labeling accuracies (from k-means or mlp classifier) determine the noisiness of the constraint in these settings

3.3. Semi-supervised Learning

Implementation Details: We train a 2-layer MLP atop the cached target features of supervised set for pseudo-labeling. The pseudo-label training is performed twice per epoch (taking approximately 40 seconds per training) and the label

Method	Epoch	Food 101	CIFAR 10	CIFAR 100	SUN 397	Cars 196	Air-craft	DTD	Pets	Calt. 101	Flwr 102	**Mean Trans**	**Linear IN-1k**
Supervised Models													
Xent	200	67.7	89.8	72.5	57.5	43.7	39.8	67.9	91.8	91.1	88.0	71.0	77.2
Xent	90	72.8	91.0	74.0	59.5	56.8	48.4	70.7	92.0	90.8	93.0	74.9	76.2
ProtoNW	200	73.3	93.2	78.3	61.5	65.0	57.6	73.7	92.2	94.3	93.7	78.3	76.0
SupCon	200	72.5	93.8	77.7	61.5	64.8	58.6	74.6	**92.5**	93.6	94.1	78.4	**77.5**
Xent	1000	72.3	93.6	78.3	61.9	66.7	61.0	**74.9**	91.5	94.5	94.7	78.9	76.3
CMSF top-*all*	200	73.7	94.2	**78.7**	62.1	**71.7**	**64.1**	73.4	**92.5**	94.5	**95.8**	**80.1**	75.7
CMSF top-10	200	**74.9**	**94.4**	**78.7**	**62.7**	70.8	63.4	73.8	92.2	**94.9**	95.6	**80.1**	76.4
Self-Supervised Models													
SimCLR	1000	72.8	90.5	74.4	60.6	49.3	49.8	**75.7**	84.6	89.3	92.6	74.0	69.3
MoCo v2	800	72.5	**92.2**	74.6	59.6	50.5	53.2	74.4	84.6	90.0	90.5	74.2	71.1
BYOL	1000	**75.3**	91.3	**78.4**	**62.2**	**67.8**	**60.6**	75.5	**90.4**	**94.2**	**96.1**	**79.2**	**74.3**
MoCo v2	200	70.4	91.0	73.5	57.5	47.7	51.2	73.9	81.3	88.7	91.1	72.6	67.5
BYOL-asym	200	70.2	91.5	74.2	59.0	54.0	52.1	73.4	86.2	90.4	92.1	74.3	69.3
MSF	200	72.3	92.7	76.3	60.2	59.4	56.3	71.7	89.8	90.9	93.7	76.3	72.1
CMSF-KM	200	**73.3**	**93.4**	**78.0**	**61.3**	**61.1**	58.0	72.7	90.2	**92.4**	94.4	**77.5**	**72.9**
CMSF-2Q	200	73.0	92.2	77.2	61.0	60.6	**58.4**	**74.1**	**91.1**	92.0	**94.5**	77.4	**73.0**

Table 2. **Transfer learning evaluation:** Our supervised CMSF model at just 200 epochs outperforms all supervised baselines on transfer learning evaluation. Our SSL model outperforms MSF, the comparable state-of-the-art approach, by 1.2 points on average over 10 datasets. We get the results for MoCo v2, MSF, and BYOL-asym from [42], SimCLR and Xent (1000 epoch) from [17], and BYOL from [32].

Method	Fine-tuned	Epochs	Top-1		Top-5	
			1%	10%	1%	10%
Supervised	✓		25.4	56.4	48.4	80.4
PIRL [51]	✓	800	-	-	57.2	83.8
CO2 [77]	✓	200	-	-	71.0	85.7
SimCLR [17]	✓	1000	48.3	65.6	75.5	87.8
InvP [74]	✓	800	-	-	78.2	88.7
BYOL [32]	✓	1000	53.2	68.8	78.4	89.0
SwAV [15]	✓	800	**53.9**	**70.2**	**78.5**	**89.9**
MoCo v2 [19]	✗	800	51.5	63.6	77.6	86.1
BYOL [32]	✗	1000	55.7	**68.6**	80.0	**88.6**
CompRess [2]	✗	1K+130	**59.7**	67.0	**82.3**	87.5
MoCo v2 [19]	✗	200	43.6	58.4	71.2	82.9
BYOL-asym	✗	200	47.9	61.3	74.6	84.7
ISD [67]	✗	200	53.4	63.0	78.8	85.9
MSF [42]	✗	200	55.5	66.5	**79.9**	87.6
CMSF-KM	✗	200	**56.5**	**67.4**	**79.9**	87.3
CMSF-2Q	✗	200	56.4	**67.5**	79.8	**87.7**

Table 3. **Evaluation on small labeled ImageNet** : We compare our model to MSF and other baselines on ImageNet 1% and 10% Linear evaluation benchmark. "Fine-tuned" refers to fine-tuning the entire backbone network instead of a single linear layer. CMSF-KM and CMSF-2Q both outperform MSF.

assignment is done in an online fashion for each mini-batch. The confidence threshold for pseudo-labeling is set to 0.85. We use the same optimizer settings as in self-supervised CMSF for the pre-training stage. Similar to S4L [86], we perform two stages of fine-tuning with supervised and pseudo-labels. We fine-tune the backbone network with two MLPs (as in PAWS [6]) on the 10% labeled set for 20 epochs and pseudo-label the train set. Samples above confidence threshold (nearly 30% of dataset) are combined with supervised set to fine-tune again for 20 epochs (more details in appendix). The second fine-tuning is equivalent to 5 epochs with full data and is a small increase in our total compute. This step is needed since we do not directly optimize cross-entropy loss in pre-training as in [57,64,82].

Evaluation: The final epoch parameters are used to perform evaluation. We report top-1 accuracy on the ImageNet validation set. We additionally report the total number of FLOPs for forward and backward passes (backward is $2\times$ forward) through ResNet-50 backbone and the number of GPUs/TPUs used by each method in the pre-training stage (more details in appendix).

Baselines: We compare the proposed approach (*CMSF-Pseudo*) with self- and semi-supervised approaches. *CMSF-Baseline* minimizes unconstrained MSF loss on the unlabeled examples (no pseudo-labeling) and CMSF loss on the labeled examples. We provide comparison of PAWS method with different support set sizes. We train PAWS on 4x 16GB GPUs with maximum possible support set size (200 classes, 2 images/class) using code provided by the authors. We also report results using mixed precision training (*CMSF-Pseudo-mix precision*) as done in PAWS [6] with batch size of 768 since mixed precision has lower memory requirement.

Method	Epochs	Batch Size	GPUs	FLOPs (x10^{17})	Top-1
Self-supervised Pre-training					
Mean Shift [42]	200	256	4	40	67.4
BYOL [32]	1000	4096	512*	399	68.8
SwAV [15]	800	4096	64	371	70.2
SimCLRv2 [18]	800	4096	128*	160	68.4
Semi-supervised Pre-training					
SimCLRv2 (+Self Dist) [18]	1200	4096	128*	200	70.5
UDA† [82]	800	15872	64*	98	68.1
FixMatch† [64]	300	6144	32*	69	71.5
MPL† [57]	800	2048	-	295	73.9
PAWS (support=6720) [6]	300	4096	64	214	75.5
PAWS (support=1680) [6]	100	256	8	145	70.2
PAWS (support=400) [6]	100	256	4	70	62.9
CMSF-Baseline	200	256	4	40	68.6
CMSF-Pseudo	200	256	4	40	69.9
CMSF-Pseudo-mix precision	200	768	4	40	70.5

Table 4. **Semi-supervised learning on ImageNet dataset with 10% labels:** FLOPs denotes the total number of FLOPS for forward and backward passes through ResNet-50 backbone while batch size denotes the sum of labeled and unlabeled samples in a batch. CMSF-Pseudo-mix precision is compute and resource efficient, achieving SOTA performance at comparable compute. PAWS requires large number of GPUs to be compute efficient and its performance drastically drops with 4/8 GPUs. † Trained with stronger augmentations like RandAugment [23]. * TPUs are used.

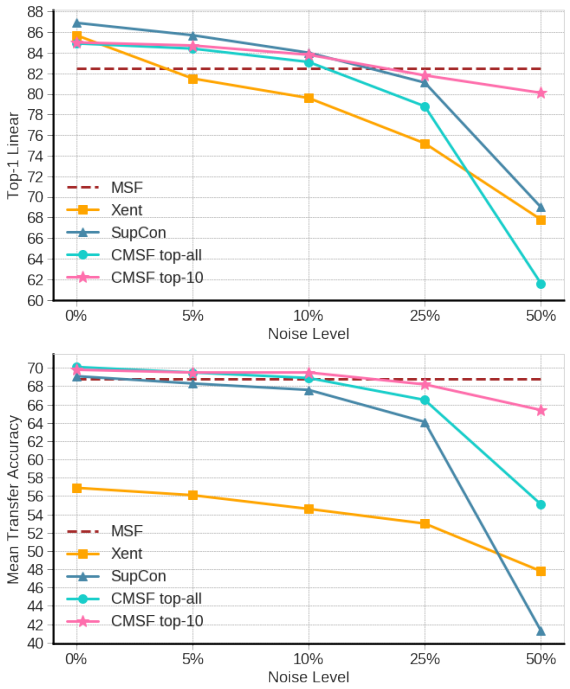

Figure 6. **Noisy supervised setting on ImageNet-100:** Our method is more robust to noisy annotation compared to Xent and SupCon. Also, using top-*all* degrades the results since all images from a single category are not guaranteed to be semantically related due to noisy labels. Mean Transfer Accuracy is average accuracy of each model over 10 transfer datasets in our settings.

3.3.1 Results

CMSF-Pseudo-mix precision achieves comparable performance to most methods with significantly less training and without the use of stronger augmentation schemes like RandAugment [23] (table 4, Fig. 1). PAWS with a support set size of 6720 outperforms other approaches. However, this requires significantly higher compute (4.8×) and resources (64 GPUs) compared to CMSF-Pseudo-mix precision (4 GPUs). Since PAWS requires a large support set, it does not scale well to lower resource (4/8 GPUs) settings even if the total compute remains the same. When trained on only 4 GPUs, CMSF outperforms PAWS by **7.6%** points. Additional ablations and results on ImageNet-100 dataset are in appendix.

4. Related Work

Self-supervised learning (SSL): Here, we want to learn representations without any annotations. One way to learn form unlabeled data is by solving a pretext task. Examples of a pretext task are colorization [89], jigsaw puzzle [54], counting [55] and rotation prediction [30] and use auxiliary information in SSL [71]. Another class of SSL methods are based on instance discrimination [24]. The idea is to classify each image as its own class. Some methods adopted the idea of contrastive learning for instance discrimination [14–17, 33]. BYOL [32] proposes a non-contrastive approach by removing the negative set from contrastive SSL methods and simply regressing target view of an image from the query view. CLD [76] integrate between instance similarities in contrastive learning. MSF [42] and MYOW [8] generalize BYOL by regressing target view and its NNs. NNCLR [25] use NN in contrastive. Our idea is adopted from MSF [42] by using an additional source of knowledge to constrain the NN search space for the target view. Thus unlike MSF, we can group far away neighbors that are semantically similar.

Supervised learning: Cross-entropy is a well known loss function for supervised learning [11, 48, 59]. Cross-entropy is contrastive in nature since ground truth probability of each class is either 0 or 1. One drawback of Cross-entropy is its lack of robustness to noisy labels [65, 90]. [52, 66, 70, 83] address the issue of hard labeling, *e.g.*, (one-hot labels) with label smoothing, [9, 27, 36] replace hard labels with prediction of pretrained teacher, and [85, 88] propose an augmentation strategy to train on combination of instances and their labels. Another line of works [31, 60] have attempted to learn representations with good kNN performance. Supervised Contrastive Learning (SupCon) [41] and [80] improve upon [31] by changing the

distance to inner product on ℓ_2 normalized embeddings. Our method is different as it does not use negative samples which makes it non-contrastive. Moreover, we focus on learning transferable representations instead of just focusing on the pretraining task. Note that the supervised setting is not the main focus of our work and we include it to better understand the effect of using constrained NNs particularly in the noisy label setting.

Semi-supervised learning: Several approaches combine self-supervised and supervised learning to form semi-supervised methods. S4L [86] uses rotation prediction based loss on the unlabeled set along with cross-entropy loss on the labeled set. Similarly, SuNCEt [5] combines SimCLR [17] and SwAV [15] methods with supervised contrastive loss. Pseudo-labeling is frequently used in semi-supervised learning. In Pseudo-Label [45], the network is trained with cross-entropy loss using supervised data on the labeled examples and pseudo-labels on the unlabeled ones. In SimCLR-v2 [18], a teacher network is pretrained using SimCLR [17] and fine-tuned with supervised labels. The teacher is then distilled to a student network using pseudo-labels on the unlabeled set. FixMatch [64] uses pseudo-labels obtained using a weakly augmented image to train a strongly augmented version of the same image. UDA [82] leverages strong data augmentation techniques in enforcing this consistency in pseudo-labels across augmentations. MPL [57] optimizes a student network using pseudo-labels from a teacher network, while the teacher is optimized to maximize the student's performance on the labeled set. PAWS [6] uses consistency based loss on soft pseudo-labels obtained in a non-parametric manner. Our method too uses pseudo-labels to train the unlabeled samples. However, we do not directly optimize with the labeled samples using cross-entropy loss but use them to obtain pseudo-labels which are used as a constraint in the MSF [42] approach.

Constrained clustering: Constrained clustering has been studied before [10, 28, 47]. [87] adds various constraints to deep k-means-like clustering. [39] uses constraints with Graph-Laplacian PCA. [4, 72] generalize mean-shift algorithm with kernel learning and pairwise constraints. Our method is a simple non-contrastive mean-shift algorithm that is inspired by the recent success in self supervised learning literature by comparing different augmentations of the same image.

Metric learning: The goal of metric learning is to train a representation that puts two instances close in the embedding space if they are semantically close. Two important methods in metric learning are: triplet loss [21, 61, 78] and contrastive loss [13, 63]. Both use positive and negative samples, but the number of negatives is larger in contrastive losses. Metric learning methods perform well on tasks like image retrieval [79] and few-shot learning [62, 73]. Prototypical networks [62] is similar to a contrastive version of our method with top-all.

5. Conclusion

MSF is a recent SSL method that pulls an image towards its nearest neighbors. We argue that the model can benefit from more diverse yet pure neighbors. Hence, we generalize MSF method by constraining the nearest neighbor search. This opens the door to using the mean-shift idea to various settings of self-supervised, supervised and semi-supervised. To construct the constraint, our SSL method uses cached augmentations from the previous epoch while the supervised and semi-supervised settings use labels or pseudo-labels. We show that our method outperforms SOTA approaches like MSF in SSL, PAWS in semi-supervised, and supervised contrastive in transfer-learning evaluation of supervised settings.

Limitations and societal impact: Recent SSL methods are usually computationally very expensive leading to worse environmental impact and also exclusion of smaller research labs. While our experiments are more efficient and accessible than most SOTA methods *e.g.*, PAWS, we limit our training length to 200 epochs due to resource constraints. We do not empirically verify that the improvements observed over SOTA approaches at lower epochs (200) are consistent with longer training (800-1000epochs). Using a large unlabeled dataset may result in models affected by unwanted biases with negative societal impact. Such concerns are actively being studied in the community and are out of the scope of this paper. Also, similar to most recent SSL methods, our method depends on augmentations which may need to be adapted for other domains, e.g, medical images.

Acknowledgment: This material is based upon work partially supported by the United States Air Force under Contract No. FA8750-19-C-0098, funding from SAP SE, and also NSF grant numbers 1845216 and 1920079. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force, DARPA, or other funding agencies.

References

[1] Torchvision models. https://pytorch.org/docs/stable/torchvision/models.html. 5, 17

[2] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Compress: Self-supervised learning

by compressing representations. *Advances in Neural Information Processing Systems*, 33, 2020. 4, 5, 7

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019. 16

[4] Saket Anand, Sushil Mittal, Oncel Tuzel, and Peter Meer. Semi-supervised kernel mean shift clustering. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1201–1215, 2013. 9

[5] Mahmoud Assran, Nicolas Ballas, Lluis Castrejon, and Michael Rabbat. Supervision accelerates pre-training in contrastive semi-supervised learning of visual representations. *arXiv preprint arXiv:2006.10803*, 2020. 9

[6] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. *ICCV*, 2021. 2, 7, 8, 9, 18

[7] Mehdi Azabou, Mohammad Gheshlaghi Azar, Ran Liu, Chi-Heng Lin, Erik C Johnson, Kiran Bhaskaran-Nair, Max Dabagia, Bernardo Avila-Pires, Lindsey Kitchell, Keith B Hengen, et al. Mine your own view: Self-supervised learning through across-sample prediction. *arXiv preprint arXiv:2102.10106*, 2021. 1

[8] Mehdi Azabou, Mohammad Gheshlaghi Azar, Ran Liu, Chi-Heng Lin, Erik C. Johnson, Kiran Bhaskaran-Nair, Max Dabagia, Bernardo Avila-Pires, Lindsey Kitchell, Keith B. Hengen, William Gray-Roncal, Michal Valko, and Eva L. Dyer. Mine your own view: Self-supervised learning through across-sample prediction, 2021. 8

[9] Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2018. 8

[10] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008. 9

[11] Eric Baum and Frank Wilczek. Supervised learning of probability distributions by neural networks. In D. Anderson, editor, *Neural Information Processing Systems*. American Institute of Physics, 1988. 6, 8

[12] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 4, 19

[13] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6:737–744, 1993. 9

[14] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 5, 8

[15] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, pages 9912–9924. Curran Associates, Inc., 2020. 5, 7, 8, 9, 18

[16] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 8

[17] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 5, 6, 7, 8, 9, 16

[18] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020. 8, 9, 18

[19] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 5, 7

[20] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020. 5

[21] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005. 9

[22] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Computer Vision and Pattern Recognition*, 2014. 4, 19

[23] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. Randaugment: Practical automated data augmentation with a reduced search space. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18613–18624. Curran Associates, Inc., 2020. 8

[24] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems*, pages 766–774, 2014. 8

[25] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations, 2021. 1, 5, 8

[26] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 2004. 4, 19

[27] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks, 2018. 8

[28] Pierre Gançarski, Bruno Crémilleux, Germain Forestier, Thomas Lampert, et al. Constrained clustering: Current and new trends. In *A Guided Tour of Artificial Intelligence Research*, pages 447–484. Springer, 2020. 9

[29] Spyros Gidaris, Andrei Bursuc, Gilles Puy, Nikos Komodakis, Matthieu Cord, and Patrick Perez. Obow: Online bag-of-visual-words generation for self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6830–6840, June 2021. 5

[30] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. 8

[31] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. *Advances in neural information processing systems*, 17:513–520, 2004. 8

[32] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 1, 2, 3, 5, 6, 7, 8, 16, 18, 19

[33] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 1, 2, 5, 8

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 17

[35] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 5

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 8

[37] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 17

[38] Tri Huynh, Simon Kornblith, Matthew R Walter, Michael Maire, and Maryam Khademi. Boosting contrastive self-supervised learning with false negative cancellation. *arXiv preprint arXiv:2011.11765*, 2020. 1

[39] Yuheng Jia, Junhui Hou, and Sam Kwong. Constrained clustering with dissimilarity propagation-guided graph-laplacian pca. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 9

[40] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 2020. 1

[41] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33, 2020. 6, 8, 16

[42] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10326–10335, October 2021. 1, 2, 3, 5, 7, 8, 9, 18, 19

[43] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *Workshop on 3D Representation and Recognition*, Sydney, Australia, 2013. 4, 19

[44] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 4, 19

[45] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013. 9

[46] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A domain-agnostic strategy for contrastive representation learning. In *International Conference on Learning Representations*, 2020. 1

[47] Pierre Legendre. Constrained clustering. *Develoments in Numerical Ecology*, pages 289–307, 1987. 9

[48] Esther Levin and Michael Fleisher. Accelerated learning in layered neural networks. *Complex systems*, 2(625-640):3, 1988. 6, 8

[49] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018. 16

[50] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 4, 19

[51] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv preprint arXiv:1912.01991*, 2019. 7

[52] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help?, 2020. 8

[53] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008. 4, 19

[54] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 8

[55] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5898–5906, 2017. 8

[56] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *Computer Vision and Pattern Recognition*, 2012. 4, 19

[57] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021. 7, 8, 9, 18

[58] Colorado J Reed, Sean Metzger, Aravind Srinivas, Trevor Darrell, and Kurt Keutzer. Selfaugment: Automatic augmentation policies for self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2674–2683, 2021. 1

[59] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 6, 8

[60] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419. PMLR, 2007. 8

[61] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 9

[62] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 6, 9, 16

[63] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865, 2016. 9

[64] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33, 2020. 7, 8, 9, 18

[65] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels, 2015. 8

[66] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015. 8

[67] Ajinkya Tejankar, Soroush Abbasi Koohpayegani, Vipin Pillai, Paolo Favaro, and Hamed Pirsiavash. Isd: Self-supervised learning by iterative similarity distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9609–9618, October 2021. 5, 7

[68] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 6

[69] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*, volume 33, pages 6827–6839. Curran Associates, Inc., 2020. 1

[70] Hugo Touvron, Alexandre Sablayrolles, Matthijs Douze, Matthieu Cord, and Hervé Jégou. Grafit: Learning fine-grained image representations with coarse labels, 2020. 8

[71] Yao-Hung Hubert Tsai, Tianqin Li, Weixin Liu, Peiyuan Liao, Ruslan Salakhutdinov, and Louis-Philippe Morency. Integrating auxiliary information in self-supervised learning, 2021. 8

[72] Oncel Tuzel, Fatih Porikli, and Peter Meer. Kernel methods for weakly supervised mean shift clustering. In *2009 IEEE 12th International Conference on Computer Vision*, pages 48–55. IEEE, 2009. 9

[73] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning, 2017. 9

[74] Feng Wang, Huaping Liu, Di Guo, and Sun Fuchun. Unsupervised representation learning by invariance propagation. In *Advances in Neural Information Processing Systems*, volume 33, pages 3510–3520. Curran Associates, Inc., 2020. 5, 7

[75] Guangrun Wang, Keze Wang, Guangcong Wang, Philip H. S. Torr, and Liang Lin. Solving inefficiency of self-supervised representation learning, 2021. 1, 5

[76] Xudong Wang, Ziwei Liu, and Stella X. Yu. Unsupervised feature learning by cross-level instance-group discrimination. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12586–12595, June 2021. 8

[77] Chen Wei, Huiyu Wang, Wei Shen, and Alan Yuille. Co2: Consistent contrast for unsupervised visual representation learning. *arXiv preprint arXiv:2010.02217*, 2020. 5, 7

[78] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006. 9

[79] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 9

[80] Zhirong Wu, Alexei A. Efros, and Stella X. Yu. Improving generalization via scalable neighborhood component analysis, 2018. 8

[81] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition*, 2010. 4, 19

[82] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *NeurIPS*, 2020. 7, 8, 9, 18

[83] Yuanhong Xu, Qi Qian, Hao Li, Rong Jin, and Juhua Hu. Weakly supervised representation learning with coarse labels, 2021. 8

[84] Asano YM., Rupprecht C., and Vedaldi A. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2020. 5

[85] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019. 8

[86] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 7, 9

[87] Hongjing Zhang, Sugato Basu, and Ian Davidson. A framework for deep constrained clustering-algorithms and advances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 57–72. Springer, 2019. 9

[88] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018. 8

[89] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 8

[90] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018. 8

Appendix

Here, we provide additional results and analysis on self-supervised (section A), semi-supervised (section B) and supervised (section C) settings. Additional results include analysis of retrieved far neighbors in various settings (Figs. A1, A2, A3) and ablations to justify various design choices (tables A1, A2, A3, A4, A5, A6, A8, A9). More details on implementation (section D) and compute calculation (Eq. 1, table A7) are provided. We also make the code available here: https://github.com/UCDvision/CMSF

A. Results on Self-supervised Setting

In Figure A5 and A4, we visualize constrained samples and their corresponding rank in the primary memory bank M. While the samples have a high rank in the primary memory bank, they are semantically related.

A.1. Effect of k' in Top-k':

In CMSF-2Q, we first calculate top-k' samples (the first k' nearest neighbors of the target) from the secondary memory bank M'. Next, we use those indices to constrain NN search space in the primary memory bank M. To explore the effect of top-k', we varied the value of k' in CMSF-2Q. Settings of this ablation is similar to our CMSF-2Q in the main experiment. Results are shown in table A1. We observe that increasing k' will decrease the accuracy of the model. As observed in Fig. A1, overlap between constrained and unconstrained NN set increases with increasing value of k'. Note that in a case where $k' = \infty$, CMSF-2Q will be exactly same as the MSF baseline.

	k'=5	k'=10	k'=20	k'=40	k'=80
NN	63.2	62.9	62.7	62.3	61.7
20-NN	66.4	66.1	65.9	65.6	65.0

Table A1. **Effect of k' in top-k' NNs sampling within M':** We constrain top-k NN search space in M with top-k' samples from M'. Increasing k' results in a drop in accuracy.

B. Results on Semi-supervised Setting

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

B.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. A3, 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. A3 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 A2. 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 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.

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**

Table A2. **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.

B.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

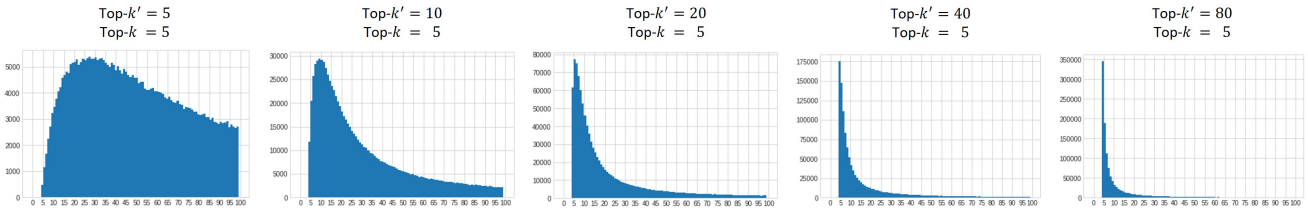

Figure A1. **Effect of k' in Top-k' (CMSF-2Q):** We plot the histogram of constrained sample ranks using checkpoint 50 of CMSF-2Q. First, we get indices of top-k' samples from M'. Then we limit Top-k NN search space on M to those indices. Overlap between constrained and unconstrained NN set increases with increasing value of k'.

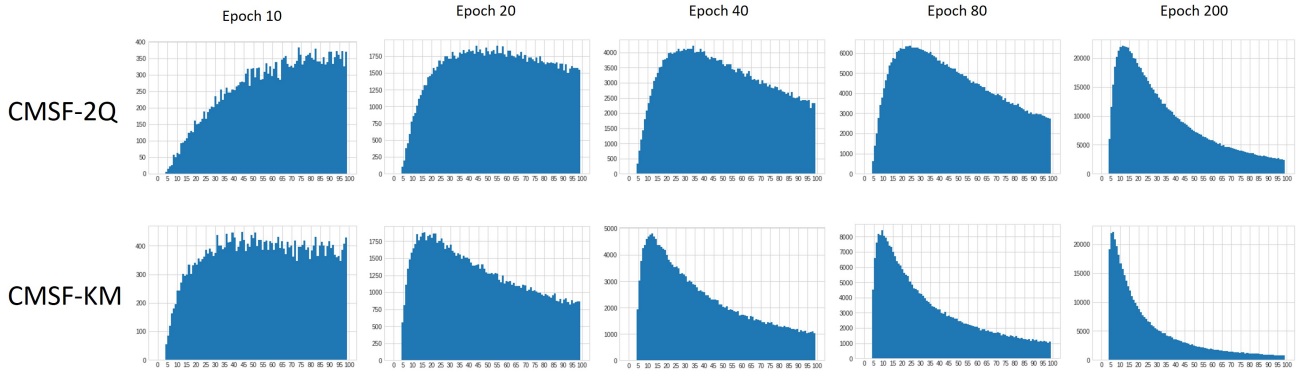

Figure A2. **Histogram of constrained samples:** We plot the histogram of constrained sample ranks in multiple stages of training of both CMSF-2Q and CMSF-KM for comparison. A large number of distant neighbors are part of constraint in the early stages of training while there is a higher overlap between constrained and unconstrained NN set towards the end of training. CMSF-2Q retrieves farther neighbors compared to CMSF-KM.

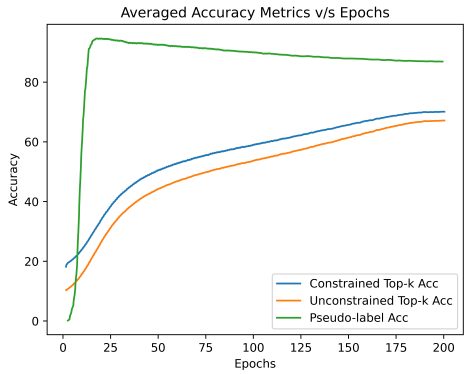

Figure A3. **Unconstrained NN accuracy in semi-supervised:** For analysis, we track the pseudo-labeling accuracy and accuracy of top-k neighbors chosen with and w/o applying the pseudo-label based constraint during training. The more accurate constrained NNs provide a better training signal. Pseudo-label accuracy on confident samples remains high throughout training, decreasing slightly as more confident samples are added.

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 advantage

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

Table A3. **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.

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 A3). 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.

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 A4. **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.

B.3. Pseudo-label Classifier Selection

The classifier used to generate pseudo-labels plays a crucial role in obtaining effective constraint sets for CMSF. 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 A4. 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 A4, 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.

B.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 A5, 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.

C. Results on Supervised Setting

C.1. 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 A6. Training and evaluation details are the same as in Section 3.2 of the main submission.

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

Table A5. **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.

D. Implementation Details

D.1. Transfer Learning

We use the LBFGS optimizer (max_iter=20, and history_size=10) along with the Optuna library [3] in the Ray hyperparameter tuning framework [49]. 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 [32] and SimCLR [17] 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.

D.2. Supervised Setting

D.2.1 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 [41], we use the first 10 epochs for learning-rate warmup. For both SupCon and ProtoNW, the temperature is 0.1.

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 [62], 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.

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		
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		
target in top-10	80.1	76.4
target not in top-10	80.3	76.4

Table A6. **Ablations of baselines and CMSF:** 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.

D.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 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 A7) 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 [34] 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 [37]. 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.

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

Method	Unlabeled			Labeled			Mini-Batch	Iters /epoch	Epochs	Total Pass ($\times 10^8$)	FLOPs ($\times 10^{18}$)
	Fwd	Bwd	BS	Fwd	Bwd	BS					
Mean Shift [42]	2	1	256				768	5004	200	7.7	40
BYOL [32]	4	2	4096				24576	312	1000	76.7	399
SwAV [15]	3.1	3.1	4096				25395	312	800	63.4	371
SimCLRv2 [18]	2	2	4096				16384	312	800	40.9	160
UDA† [82]	2	1	15360	1	1	512	47104	40000		18.8	98
FixMatch† [64]	2	1	5120	1	1	1024	17408	250	300	13.1	69
MPL† [57]	3	2	2048	2	2	128	10752	500000		53.8	295
PAWS (support=6720) [6]	3.1	3.1	4096	1	1	6720	38835	312	300	36.6	214
PAWS (support=1680) [6]	3.1	3.1	256	1	1	1680	4947	5004	100	24.8	145
PAWS (support=400) [6]	3.1	3.1	256	1	1	400	2387	5004	100	12.0	70
CMSF-Baseline	2	1	256				768	5004	200	7.7	40
CMSF-Pseudo	2	1	256				768	5004	200	7.7	40
CMSF-Pseudo-mix precision	2	1	768				2304	1668	200	7.7	40

Table A7. **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.

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 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 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 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 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 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 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 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 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 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 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 A8. **Noisy supervised setting on ImageNet-100:** Our method is more robust to noisy annotation compared to Xent and SupCon. Also, using top-*all* results in degradation since all images from a single category are not guaranteed to be semantically related.

Dataset	Classes	Train samples	Val samples	Test samples	Accuracy measure	Test provided
Food101 [12]	101	68175	7575	25250	Top-1 accuracy	-
CIFAR-10 [44]	10	49500	500	10000	Top-1 accuracy	-
CIFAR-100 [44]	100	45000	5000	10000	Top-1 accuracy	-
Sun397 (split 1) [81]	397	15880	3970	19850	Top-1 accuracy	-
Cars [43]	196	6509	1635	8041	Top-1 accuracy	-
Aircraft [50]	100	5367	1300	3333	Mean per-class accuracy	Yes
DTD (split 1) [22]	47	1880	1880	1880	Top-1 accuracy	Yes
Pets [56]	37	2940	740	3669	Mean per-class accuracy	-
Caltech-101 [26]	101	2550	510	6084	Mean per-class accuracy	-
Flowers [53]	102	1020	1020	6149	Mean per-class accuracy	Yes

Table A9. **Transfer dataset details:** Train, val, and test splits of the transfer datasets are listed in this table. **Test split:** We follow the details in [42]. 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 [32], 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.

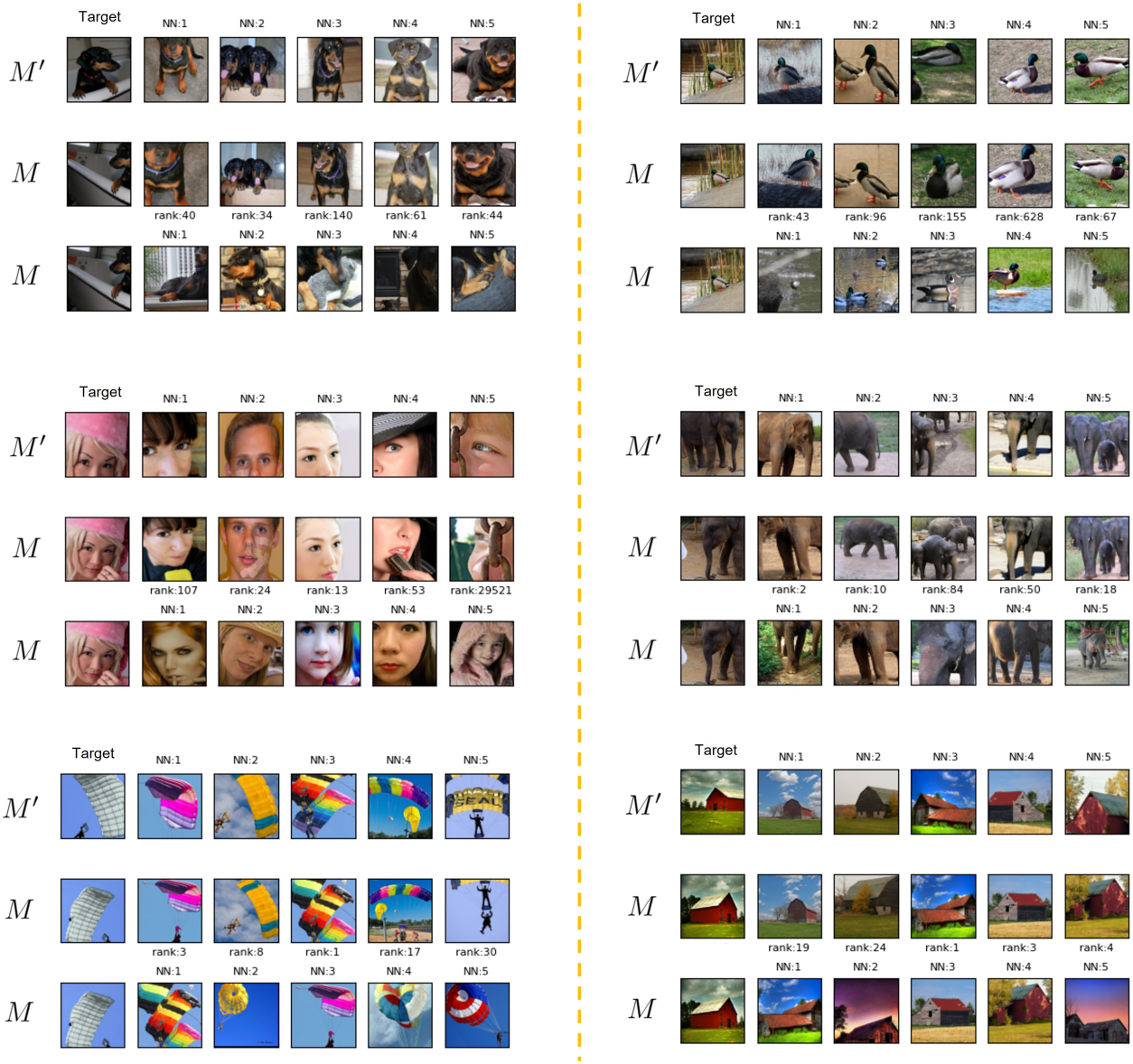

Figure A4. **CMSF-2Q Nearest neighbor selection :** We use epoch 100 of CMSF-2Q to visualize Top-5 NN from constrained and unconstrained memory bank. First row is NNs from the second memory bank M', that is exactly the same as M but containing a different augmentation. Samples of the second row are NNs from second memory bank M' in M, therefore they are different augmentations of first row. Additionally, We show their rank in M as well. The last row is NNs from the first memory bank M. Note that constrained samples in M (second row), have high rank while they are semantically similar to the target.

Figure A5. **Nearest neighbor selection on constrained memory bank:** We visualize more samples of Figure 3 without any cherry-picking. We show results of three different checkpoints of CMSF-KM.