

Supplementary: On Partial Prototype Collapse in the DINO Family of Self-Supervised Methods

BMVC 2024 Submission # 949

A Appendix

A.1 Extended related work

Clustering-based self-supervised learning: Self-supervised learning based on the clustering pretext task is a promising paradigm that has proven to be successful and grown tremendously in recent years. Initial works [1, 2, 3] used a two-stage process of assigning pseudo-labels by clustering the representations and then training the representations using the pseudo-labels as targets. Caron et al. [4] proposed uniform pseudo-label sampling that is equivalent to weighting the loss contribution of an input by the inverse of its assigned cluster's size. Caron et al. [5] used online assignment of pseudo-labels in every batch by clustering the representations over a small window of batches. By adapting this to ViTs, Caron et al. [6] proposed a self-distillation framework where a teacher network produced the target latent classes. Govindarajan et al. [7] demonstrated that this objective corresponds to learning a von Mises-Fisher mixture distribution. Li et al. [8] extended the DINO objective to patch tokens while also leveraging efficient architectures like Swin [20]. iBOT [60] is a recent state-of-the-art method that poses the masked image modeling (MIM) task of BeIT [9] as a clustering task. Another branch of works have focused on improving the few-shot learning performance of these methods [1, 24]. Recently, DINOv2 [25] built upon iBOT by making several modifications. By pre-training on the large LVD142M dataset, DINOv2 demonstrated performance surpassing many state-of-the-art visual benchmarks at image and pixel levels.

A.2 Sinkhorn-Knopp and probability centering

Let the batch of B logit scores be denoted as $\mathbf{L} \in \mathbb{R}^{B \times K}$ with corresponding probability distributions \mathbf{P} . Then, the Sinkhorn-Knopp adjusted probability distributions $\tilde{\mathbf{P}}$ are obtained by alternating between normalizing the rows and columns of the matrix $\exp(\mathbf{L})$, so that they sum up to 1. Note that the exponent function is applied element-wise to the matrix. Let the elements of the matrix be denoted as $\mathbf{L}_{b,k}$. Then, normalizing along the rows yields,

$$\tilde{\mathbf{P}}_{b,k} \leftarrow \frac{\exp(\mathbf{L}_{b,k})}{\sum_b \exp(\mathbf{L}_{b,k})} = \frac{\frac{1}{B} \exp(\mathbf{L}_{b,k})}{\frac{1}{B} \sum_b \exp(\mathbf{L}_{b,k})} = \frac{1}{B} \exp(\mathbf{L}_{b,k} - \log(\frac{1}{B} \sum_b \exp(\mathbf{L}_{b,k}))).$$

Next, normalizing $\tilde{\mathbf{P}}$ along the columns we obtain,

$$\tilde{\mathbf{P}}_{b,k} \leftarrow \frac{\exp(\mathbf{L}_{b,k} - \log(\frac{1}{B} \sum_b \exp(\mathbf{L}_{b,k})))}{\sum_{j=1}^K \exp(\mathbf{L}_{b,j} - \log(\frac{1}{B} \sum_b \exp(\mathbf{L}_{b,j})))}.$$

If we consider the initial logit scores \mathbf{L}_b to be already normalized over the components K such that $\sum_k \exp(\mathbf{L}_{b,k}) = 1$, then the exponents within the inner sum can be replaced with probabilities. Thus, we obtain the probability distributions after 1 iteration of Sinkhorn-Knopp adjustment as,

$$\tilde{\mathbf{P}}_{b,k}^{(\text{sk1})} \leftarrow \frac{\exp(\mathbf{L}_{b,k} - \log(\frac{1}{B} \sum_b \mathbf{P}_{b,k}))}{\sum_{j=1}^K \exp(\mathbf{L}_{b,j} - \log(\frac{1}{B} \sum_b \mathbf{P}_{b,j}))}.$$

On the other hand, the probability centered distributions proposed by Govindarajan et al. [13] are obtained as follows, where the centering parameter c_k is calculated as a moving average estimate with momentum parameter m :

$$\tilde{\mathbf{P}}_{b,k}^{(\text{pc})} = \frac{\exp(\mathbf{L}_{b,k} - c_k)}{\sum_{j=1}^K \exp(\mathbf{L}_{b,j} - c_j)}, \quad c_k \leftarrow mc_k + (1 - m) \log \left[\frac{1}{B} \sum_{b=1}^B \mathbf{P}_{b,k} \right].$$

Comparing the above expressions for $\tilde{\mathbf{P}}_{b,k}^{(\text{sk1})}$ and $\tilde{\mathbf{P}}_{b,k}^{(\text{pc})}$ (Eq. (2) and Eq. (3) in section 3 of the main paper), we observe that probability centering is equivalent to one iteration of Sinkhorn-Knopp with the key distinction that the logit adjustment is calculated as a moving average instead of a batch estimate.

A.3 KoLeo prototypes implementation

Given a set of K prototypes $\mathbf{W} \in \mathbb{R}^{K \times D}$, to compute the KoLeo estimate of the differential entropy of the prototypes $h_{\text{kl}}(\mathbf{W})$, we require computing nearest neighbor distances for each of the prototypes. This can be memory intensive when a large number of prototypes are used. Note that this is not a problem in the case of DINOv2 [12], as the KoLeo objective is computed between the B data representations in the batch (B is typically much smaller than K). Instead, we resort to a stochastic estimate when calculating the loss objective in each batch. For each batch, we randomly partition the prototypes into disjoint partitions containing 2048 prototypes each, $\mathbf{W} = \{\mathbf{W}_1, \dots, \mathbf{W}_T\}$, $\mathbf{W}_t \in \mathbb{R}^{2048 \times D}$. Then, we compute the KoLeo estimate as follows: $h_{\text{kl}}(\mathbf{W}) = \sum_{t=1}^T h_{\text{kl}}(\mathbf{W}_t)$. This efficient batched implementation adds negligible computational overhead, in terms of both memory and time (15 MB additional GPU memory when $K = 8192$ and unchanged image throughput).

A.4 Experimental details

A.4.1 Hyperparameter settings

The complete hyperparameter configuration for full-scale iBOT-vMF pre-trainings on ImageNet using ViT-Small/16 and ViT-Base/16 models are provided in Table A.4.1. For pre-training on iNaturalist-2018, we use a similar hyperparameter configuration except that we use pre-train both ViT-Small/16 and ViT-Base/16 models for 300 epochs. The complete hyperparameter configurations for MSN and PMSN pre-trainings on the iNaturalist-2018 dataset using the ViT-Small/16 model are provided in Table A.4.1.

Hyper-parameter	ViT-Small/16	ViT-Base/16
training epochs	800	400
batch size	1024	512
learning rate	$2e-3$	$1.5e-3$
warmup epochs	10	10
freeze last layer epochs	1	3
min. learning rate	$1e-6$	$2e-6$
weight decay	$0.04 \rightarrow 0.4$	$0.04 \rightarrow 0.4$
stochastic depth	0.1	0.1
gradient clip	3.0	0.3
optimizer	adamw	adamw
shared head	✓	✓
fp16	✓	✓
momentum	$0.996 \rightarrow 1.0$	$0.996 \rightarrow 1.0$
global crops	2	2
global crops scale	$[0.25, 1.0]$	$[0.32, 1.0]$
local crops	10	10
local crops scale	$[0.05, 0.25]$	$[0.05, 0.32]$
head mlp layers	3	3
head hidden dim.	2048	2048
head bottleneck dim.	256	256
norm last layer	✗	✗
num. prototypes	8192	8192
vmf normalization	✓	✓
centering	probability	probability
koleo reg. strength	0.1	0.1
teacher temp.	$0.04 \rightarrow 0.07$	$0.04 \rightarrow 0.07$
temp. warmup epochs	30	50
student temp.	0.1	0.1
pred. ratio	$[0.0, 0.3]$	$[0.0, 0.3]$
pred. ratio variance	$[0.0, 0.2]$	$[0.0, 0.2]$
pred. shape	block	block

Table A1: Hyperparameter settings for iBOT

A.4.2 MSN and PMSN discussion

When pre-training on the iNaturalist-2018 dataset using the ViT-Small/16 model, we run hyperparameter sweeps to select suitable values for the KL penalty strength parameter λ . We consider the values $\{1.0, 5.0, 15.0\}$. Based on the linear probing results shown in Table A.4.2, we select $\lambda = 1.0$ for MSN and $\lambda = 5.0$ for PMSN. Using a higher λ with MSN strongly encourages the MLCD to match a uniform prior distribution. When the pre-training dataset is naturally long-tailed, strongly encouraging a uniform prior leads to worse performance. However, we find a smaller penalty strength helps MSN to even outperform PMSN. This indicates that using a weak uniform prior can still be a reasonable choice when pre-training on long-tailed datasets.

A.4.3 Transfer linear probing

We perform our transfer linear classification experiments on the standard suite of datasets used in self-supervised learning: Caltech101 [19], CIFAR10, CIFAR100 [10], DTD [10], Flowers [10], Food [9], Pets [23] and SUN397 [28]. We follow the evaluation protocol from Ericsson et al. [10] and Chen et al. [9] and train L^2 -regularized linear classifiers. We select the regularization strength among a set of 45 values spaced linearly in the range $[-6, 5]$ in log-space and report the standard evaluation metric for each dataset.

Hyper-parameter	MSN	PMSN
training epochs	300	300
batch size	1536	1536
learning rate	$6e-3$	$6e-3$
warmup epochs	15	15
min. learning rate	$1e-6$	$2e-6$
weight decay	$0.04 \rightarrow 0.4$	$0.04 \rightarrow 0.4$
stochastic depth	0.1	0.1
gradient clip	3.0	3.0
optimizer	adamw	adamw
fp16	\times	\times
momentum	$0.996 \rightarrow 1.0$	$0.996 \rightarrow 1.0$
random crops	1	1
local crops	10	10
patch drop rate	0.15	0.15
head mlp layers	3	3
head hidden dim.	2048	2048
head bottleneck dim.	256	256
norm last layer	\checkmark	\checkmark
num. prototypes	8142	8142
kl penalty weight (λ)	1.0	5.0
teacher temp.	0.025	0.025
sinkhorn teacher	\checkmark	\checkmark
temp. warmup epochs	30	50
student temp.	0.1	0.1

Table A2: Hyperparameter settings for MSN / PMSN

Method	K	M	Overall	Head	Middle	Tail
<i>ViT-Small/16</i>						
MSN ($\lambda = 1$)	8142	3363	43.8	51.4	43.9	41.8
MSN ($\lambda = 5$)	8142	3123	42.3	49.6	42.5	40.4
MSN ($\lambda = 15$)	8142	1562	40.9	49.5	40.6	39.1
<i>PMSN</i>						
PMSN ($\lambda = 1$)	8142	2919	41.4	48.9	41.3	39.7
PMSN ($\lambda = 5$)	8142	3005	41.8	50.2	41.9	39.7
PMSN ($\lambda = 15$)	8142	2927	41.0	49.1	41.4	38.7

Table A3: iNaturalist-2018 linear probing accuracy with full data

A.4.4 Sinkhorn-Knopp and mean entropy maximization hyperparameters

For Sinkhorn-Knopp, we firstly use the vMF normalized version of iBOT and ablate over the number of iterations 1, 3, 5 and find that 3 iterations to work best. This choice for the number of iterations is in agreement with DINOv2 [22]. For both SK (iter=3) and mean entropy maximization and for each compute budget (2, 4 or 8 GPUs for 2 days) we ablate over the following hyperparameters:

- vMF normalization: True / False [23]
- Teacher temperature:
 - $\tau = 0.04 \rightarrow 0.07$ (default in Zhou et al. [30] and Govindarajan et al. [24])
 - $\tau = 0.05 \rightarrow 0.025$ (default in Ruan et al. [24])

For SK(iter=3), we find smaller teacher temperatures to be beneficial as in Ruan et al. [24] and using vMF normalization or not has marginal impact on the performance. For ME-MAX, we find that not using vMF normalization and a smaller teacher temperature leads to better performance.

A.4.5 Fine-tuning recipes

ImageNet fine-tuning: We fine-tune on the ImageNet dataset by following the fine-tuning recipe used in BeIT [8] and iBOT [50], which is found to produce consistently good performance in reasonably fewer epochs compared to other fine-tuning recipes. We fine-tune ViT-Small and ViT-Base models for 200 and 100 epochs respectively and use a batch size of 1024. We use a layer-wise learning rate decay of 0.75 for ViT-Small and 0.65 for ViT-Base. We report the best performance achieved after considering 4 different learning rates: $\{8e-4, 9e-4, 1e-3, 2e-3\}$.

iNaturalist-2018 fine-tuning: We find the fine-tuning recipe of DeiT [25] using a smaller learning rate and a larger number of epochs to work better for the iNaturalist-2018 dataset. This is similar to the transfer fine-tuning setup of iBOT [50]. We use a fine-tune both ViT-Small and ViT-Base models for 360 epochs using a batch size of 1024. We use learning rates of $5e-5$ and $7.5e-6$ for ViT-Small and ViT-Base respectively.

A.5 Additional results

A.5.1 MLCD regularization

To study the MLCD regularization, we focus on iBOT, which is a strong recent baseline among the DINO family of methods and also used as the foundation for DINOv2 [22]. We pre-train the models on the ImageNet-1K dataset [11] by modifying the public codebase of iBOT. We use the same hyperparameter settings as in iBOT for different ViT backbones (refer A.4.1 for details) and use the vMF normalized variants [13], which are shown to produce stable trainings and improved performance.

Here, we run ablation experiments to select the method to regularize MLCD. We pre-train ViT-S/16 backbone with different MLCD regularization techniques - Sinkhorn-Knopp (SK), probability centering (PC) and mean entropy maximization (ME-MAX). For PC, we use the vMF normalized version of iBOT. For SK and ME-MAX, we chose to use a smaller teacher temperature based on a hyperparameter search (refer A.4.1 for details). We also consider three different compute budgets (2, 4 and 8 GPUs for 2 days), which allows us to evaluate the impact of batch size on these techniques. With more GPUs, we can accommodate a larger batch size. The number of epochs is adjusted such that the total number of iterations are the same for all the compute budgets.

We do this to avoid the expensive process of optimizing the learning rates for each compute budget and regularization method. Overall, from Figure A1, we find that probability centering performs better than the other alternatives at different compute budgets. Interestingly, PC achieves performance on par or better than the alternatives, even at half of the compute budget (e.g. PC/4GPUs vs ME-MAX/8GPUs).

The methods discussed above have all been proposed in the literature as ways to regularize the MLCD. We argue that the main difference between them is whether the regularization is done over a single batch (SK, ME-MAX) or based on moving average statistics (PC). We

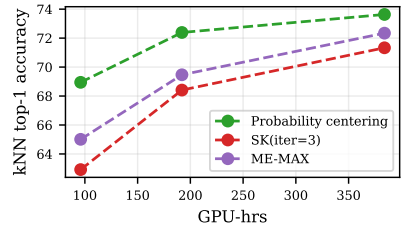


Figure A1: ImageNet top-1 kNN accuracy with different MLCD regularizations. Probability centering performs better than SK and ME-MAX at different compute budgets.

observe that PC performs significantly better than the alternatives at the lowest compute budget, which uses a small batch size. As we increase the compute budget and thereby also the batch size, the gap is reduced. This indicates that PC is more robust to the choice of batch size. We conjecture that this is due to too noisy estimates of the MLCD when computed over a batch, which is not surprising considering that we estimate probability vectors in a high-dimensional space. Therefore, in all the experiments in the main paper, we use the vMF normalized iBOT with MLCD regularized using probability centering.

A.5.2 Partial prototype collapse in more existing models

In addition to investigating partial prototype collapse in Table 1, we also investigate other self-supervised clustering methods that use a prototypical formulation such as EsViT [18] and SWaV [9]. We demonstrate in Table A4 that partial prototype collapse also occurs in these methods. We observe that partial prototype collapse also occurs in methods using Resnet50 [14], ViL [29] and CvT [27] backbones. Though we focus on ViT backbone models in this work, note that partial prototype collapse is not only limited to ViT backbones.

Backbone	Method	Initialized prototypes (K)	Unique prototypes (M)
ViT-S/16	DINO	65536	1078
ViT-B/16	DINO	65536	804
ViT-S/16	DINO-vMF	65536	1157
ViT-B/16	DINO-vMF	65536	939
ViT-S/16	iBOT	8192	3242
ViT-B/16	iBOT-vMF	8192	1170
ViT-L/16	iBOT	8192	969
ViT-L/16	iBOT**	8192	1037
Resnet50	SWaV	3000	1669
Resnet50	DINO	60000	984
Swin-Tiny/W=7	EsViT	65536	1157
Swin-Base/W=14	EsViT	65536	4088
ViL	EsViT	65536	1741
CvT	EsViT	65536	1178

Table A4: Number of unique prototypes in existing models with $\epsilon = 0.025$ (default pre-training: ImageNet-1K, **: ImageNet-22K)

A.5.3 Ablation experiment for KoLeo-prototype regularization strength

We conduct an ablation experiment to evaluate the impact of the regularization strength (λ) of the KoLeo-proto regularization term. We consider a 100 epoch iBOT-vMF pre-training using 8192 prototypes on the Imagenet dataset and evaluate $\lambda = \{0.02, 0.1, 0.5\}$. From Table A5, we find that too small $\lambda = 0.02$ is unable to fully utilize all the initialized prototypes. We observe improved performance and effective utilization of the prototypes using $\lambda = 0.1$ but do not observe further improvements from increasing λ further. The main goal of this regularization is to effectively utilize the prototypes. We use the minimum regularization strength $\lambda = 0.1$ which is sufficient to achieve this in the experiments in this paper.

A.5.4 Computational analysis

The prototype layer in the self-supervised clustering methods that use a prototypical formulation noticeably contributes to the computational cost of training such methods. The

λ	Initialized prototypes (K)	Unique prototypes (M)	kNN top-1 accuracy
0.0	8192	1045	72.39
0.02	8192	4693	72.56
0.1	8192	8192	72.62
0.5	8192	8192	72.64

Table A5: Ablation experiment for KoLeo-proto regularization strength (λ)

Batch size (B)	Number of prototypes (K)	GPU memory (GB)
64	1024	12.9
64	2048	13.7
64	4096	15.3
64	8192	18.6
64	10240	20.2
100	1024	19.7
100	2048	21.0
100	4096	23.5
100	8192	28.6
100	10240	31.1

Table A6: Computational cost of training iBOT method with different number of prototypes

weights associated with K prototypes consists of a $K \times D$ matrix. Typically, the bottleneck dimension $D = 256$. The DINO models use a large $K = 65536$ and the prototype layer alone adds an additional 16M trainable parameters to the method. A batch of size B , results in the computation of probability distributions of size $B \times K$. For iBOT, which computes the probability distributions for all tokens resulting even larger set of probability distributions of size $B \times T \times K$. The number of prototypes in iBOT is set to 8192 in the default configuration. Computing such large probability distributions involve heavy memory GPU usage and longer training times. For the default configurations of iBOT with ViT-S/16 backbone, we test the GPU memory use for different numbers of prototypes and batch sizes in the fp16 mode and report the results in Table A6. Consequently, effective utilization of prototypes can help in reducing the GPU memory required for training such models. For instance, at a batch size of 100, effectively utilizing only 1024 prototypes is significantly cheaper (19.7 GB GPU memory) than using only ~ 1024 unique prototypes out of 8192 initialized prototypes (28.6 GB GPU memory). Effective prototype utilization consumes $\sim 31\%$ lower GPU memory compared to the baseline.

A.5.5 ImageNet pre-training with a CNN backbone

In order to explore an additional method and backbone combination, we consider the DINO method pre-training using a Resnet50 backbone. We base our pre-training settings on the hyperparameter configuration in the publicly available DINO codebase¹. We use the vMF normalized version, use probability centering, 8192 prototypes and train for 100 epochs. In Table A7, we observe that the KoLeo-proto regularization mitigates the partial prototype collapse and achieves improved performance compared to the baseline and KoLeo-data regularization.

¹https://dl.fbaipublicfiles.com/dino/dino_resnet50_pretrain/args.txt

Method	Epochs	M	kNN	Linear
DINO-vMF	100	684	59.1	69.6
DINO-vMF (kd)	100	1373	59.8	70.4
DINO-vMF (kp)	100	8192	60.1	70.8

Table A7: ImageNet classification with full data (kNN, linear) using Resnet50 backbone model

Method	Cal101	C10	C100	DTD	Flwrs.	Food	Pets	SUN	Avg.
<i>ViT-Base/16</i>									
DINO-vMF	94.5	97.1	86.3	74.8	95.7	82.5	94.6	68.7	86.8
iBOT-vMF	95.5	98.0	88.0	74.7	94.8	83.6	93.9	70.2	87.3
iBOT-vMF (kp)	94.6	96.5	84.1	74.3	95.6	83.6	94.0	69.7	86.6
MSN	92.8	96.9	85.3	73.7	92.8	80.0	93.9	66.8	85.3
<i>ViT-Small/16</i>									
DINO-vMF	93.7	96.0	83.9	74.1	95.0	80.1	93.9	66.6	85.4
iBOT-vMF	94.1	96.7	84.6	72.8	94.3	80.3	94.1	67.3	85.5
iBOT-vMF (kp)	94.5	96.7	83.9	73.7	94.4	80.5	93.7	67.5	85.6
MSN	93.1	95.9	82.9	72.0	93.3	77.8	92.8	65.5	84.1
WE-SSL	94.6	93.8	81.4	74.9	93.9	79.1	92.8	66.5	84.6

Table A8: Transfer learning classification accuracies when pre-trained on Imagenet-1K and transferred to other datasets

A.5.6 Detailed transfer learning results

We follow the transfer learning protocol explained in A.4.3 to evaluate the representations learned by pre-training on Imagenet-1K and iNaturalist-2018 datasets. We report the transfer learning performance in Table A8 and Table A9 for a suite of datasets on their available validation/test splits. With Imagenet pre-training, we observe that effective utilization of the prototypes using KoLeo-proto regularization produces on par or worse transfer learning performance (based on the overall average) compared to the baseline and KoLeo-data regularization. On the other hand, with iNat-18 pre-training, we observe that effective utilization of the prototypes produces better transfer learning performance (based on the overall average) compared to the baseline and KoLeo-data regularization.

A.5.7 Representation robustness evaluation

We evaluate the impact of effective prototype utilization on representation robustness by evaluating on standard robustness benchmarks such as Imagenet-A [16] and Imagenet-C [15] datasets. The Imagenet-A dataset contains a set of adversarial images curated from the web, that commonly fool classifiers trained on Imagenet-1K dataset in a supervised manner. Imagenet-C contains images from Imagenet-1K with several types of corruptions and perturbations. We report the robustness metrics in Table A10. We observe small but consistent improvement in the robustness to adversarial and corrupted images with effective utilization of prototypes, when compared to the iBOT-vMF baseline.

A.6 Visual explanations

In this section, we present a qualitative comparison of a model trained with and without KoLeo-proto regularization. We compare the iBOT-vMF baseline method based on the ViT-S/16 backbone trained on the iNat18 dataset. We visualize the unique prototypes along with their redundancy factors in Figure A2 using t-SNE plots [26]. This illustrates the partial

Method	Cal101	C10	C100	DTD	Flwrs.	Food	Pets	SUN	Avg.
<i>ViT-Small/16</i>									
DINO-vMF	79.2	86.3	68.5	65.4	93.0	66.1	66.9	48.0	71.7
iBOT-vMF	80.3	87.0	69.2	66.3	93.6	66.4	64.2	47.4	71.8
iBOT-vMF (kd)	79.3	86.3	68.1	64.7	94.2	66.4	65.4	47.6	71.5
iBOT-vMF (kp)	80.1	86.7	69.9	66.8	93.2	66.6	65.5	47.4	72.0
MSN ($\lambda = 1$)	70.8	82.3	62.8	64.0	88.6	61.5	60.3	45.3	67.0
PMSN ($\lambda = 5$)	71.2	81.3	62.6	62.1	88.4	60.7	58.1	44.1	66.1
<i>ViT-Base/16</i>									
iBOT-vMF (kd)	82.4	87.9	70.8	66.3	94.6	68.6	66.9	48.5	73.2
iBOT-vMF (kp)	82.0	88.7	72.1	66.3	94.7	68.7	68.1	48.7	73.7

Table A9: Transfer learning classification accuracies when pre-trained on iNat-2018 and transferred to other datasets

Method	Backbone	Pre-training data	INet-A \uparrow	INet-C \downarrow
DINOv2 \ddagger	ViT-S/14	LVD-142M	33.5	54.4
DINOv2 \ddagger	ViT-B/14	LVD-142M	55.1	42.7
iBOT	ViT-L/16	INet-22K	41.5	43.9
DINO	ViT-B/8	INet-1K	23.9	56.6
iBOT-vMF	ViT-B/16	INet-1K	22.7	43.8
iBOT-vMF (kp)	ViT-B/16	INet-1K	23.5 ^(+0.8)	43.7 ^(-0.1)
iBOT-vMF	ViT-S/16	INet-1K	14.2	51.1
iBOT-vMF (kp)	ViT-S/16	INet-1K	14.3 ^(+0.1)	50.9 ^(-0.2)

\ddagger : Distilled from a larger ViT-g/14 model pre-trained using DINOv2

Table A10: Representation robustness evaluation on standard robustness benchmarks. We report the accuracy (%) for Imagenet-A dataset and mean corruption error in % (lower is better) for Imagenet-C dataset.

prototype collapse in the baseline and the impact of adding the KoLeo-proto regularization on the prototypes. KoLeo-proto regularization encourages diverse prototypes by spreading them out in the latent space, resulting in a higher number unique prototypes compared to the baseline. We visualize the representations corresponding to images that are assigned to a set of latent classes by the iBOT-vMF baseline in Figure A3. In Figure A4, we visualize the representations corresponding to the exact same images based on the iBOT-vMF model trained with KoLeo-proto regularization. We observe that the KoLeo-proto regularization encourages more fine-grained clusters compared to the baseline. In Figure A7 and Figure A6, we show a few example images belonging to the latent classes shown in Figure A5. Without KoLeo-proto regularization, only one coarse latent class is learned containing images of ducks. With KoLeo-proto regularization, this is further divided into three finer latent classes. This demonstrates that the model learns more informative representations which enable it to discriminate between these finer latent classes.

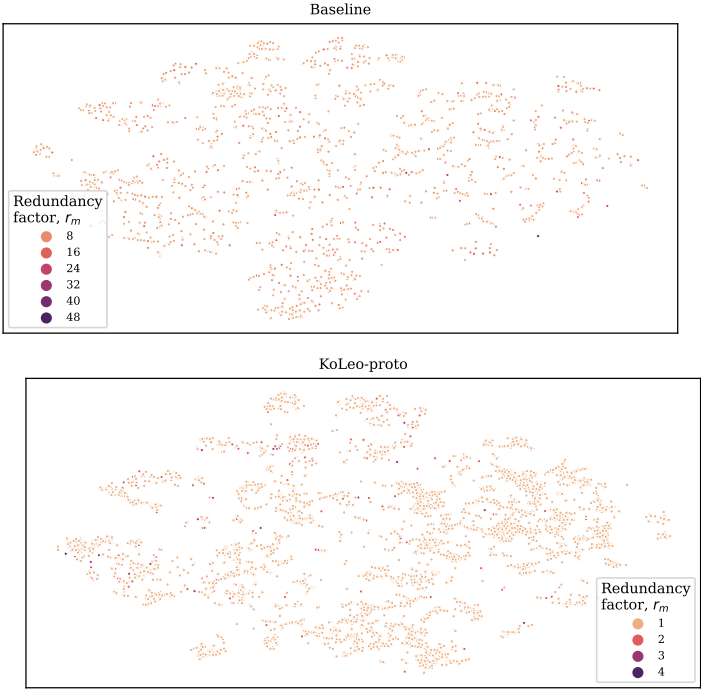


Figure A2: t-SNE plot of the M unique prototypes learned by the baseline method and with KoLeo-proto regularization, colored by their redundancy factors r_m . There are fewer unique prototypes in the baseline ($M = 1806$), noticeable from their sparse spread in the plot. The baseline prototypes are impacted by partial prototype collapse, resulting in high redundancy factors. With KoLeo-proto regularization, the model learns more unique prototypes ($M = 7895$) with significantly smaller redundancy factors compared to the baseline. With KoLeo-proto regularization, the method learns diverse prototypes that are well spread over the latent space.

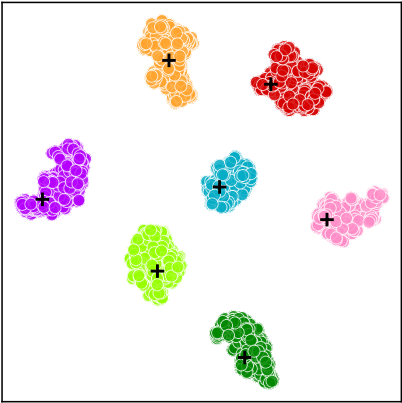


Figure A3: iBOT-vMF baseline

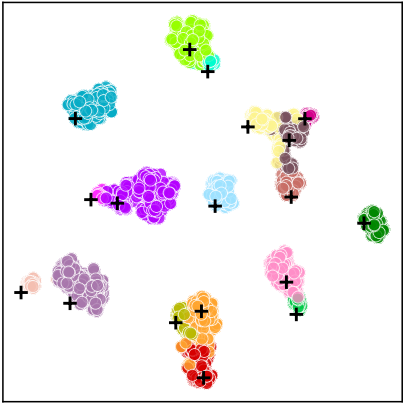


Figure A4: iBOT-vMF with KoLeo-proto

Figure A5: For the exact same set of images, the representations after the head (256 dimensional) are visualized using TSNE plots. The points are colored based on the latent class that they belong to and the corresponding prototypes are denoted using the + marker (the prototype markers are slightly shifted to prevent them from blocking some smaller clusters). The images belong to 7 latent classes in the iBOT-vMF baseline and the same images belong to 18 latent classes when the KoLeo-proto regularization is used. Partial prototype collapse in the baseline results in fewer unique prototypes and coarser clusters. KoLeo-proto regularization encourages diverse prototypes which leads to a more fine-grained clustering of the same data.



Figure A6: Sample images from the latent classes shown in Figure A4 obtained from iBOT-vMF with KoLeo-proto regularization. Same colors are used to indicate the latent classes.

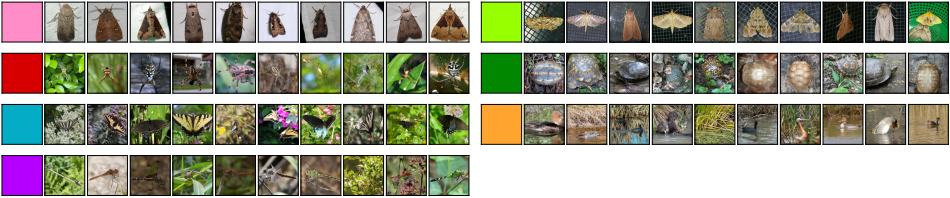


Figure A7: Sample images from the latent classes shown in Figure A3 obtained from iBOT-vMF baseline method. Same colors are used to indicate the latent classes.

References

- [1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020.
- [2] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *ECCV*, 2022.
- [3] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *ICLR*, 2022.
- [4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.
- [5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.
- [6] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019.
- [7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.
- [8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.
- [9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.
- [10] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [12] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *CVPR*, 2021.
- [13] Hariprasath Govindarajan, Per Sidén, Jacob Roll, and Fredrik Lindsten. DINO as a von mises-fisher mixture model. In *ICLR*, 2023.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [15] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.
- [16] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021.

- [17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 598
599
600
- [18] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. In *ICLR*, 2022. 601
602
603
- [19] Fei-Fei Li, Marco Andreeto, Marc’Aurelio Ranzato, and Pietro Perona. Caltech 101, 2022. URL <https://data.caltech.edu/records/20086>. 604
605
606
- [20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 607
608
609
- [21] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 610
611
612
613
- [22] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 614
615
616
617
- [23] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 618
619
620
- [24] Yangjun Ruan, Saurabh Singh, Warren Richard Morningstar, Alexander A Alemi, Sergey Ioffe, Ian Fischer, and Joshua V Dillon. Weighted ensemble self-supervised learning. In *ICLR*, 2023. 621
622
623
- [25] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 624
625
626
627
- [26] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008. 628
629
630
- [27] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 2021. 631
632
- [28] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 633
634
- [29] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *ICCV*, 2021. 635
636
637
638
- [30] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. Image BERT pre-training with online tokenizer. In *ICLR*, 2022. 639
640
641
642
643