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

Hariprasath Govindarajan[1,2]
hargov@qti.qualcomm.com
Per Sidén[2]
psiden@qti.qualcomm.com
Jacob Roll[2]
jroll@qti.qualcomm.com
Fredrik Lindsten[1]
fredrik.lindsten@liu.se

[1] Department of Computer and Information Science, Linköping University, Sweden
[2] Arriver Sweden Software AB

Abstract

A prominent self-supervised learning paradigm is to model the representations as clusters, or more generally as a mixture model. Learning to map the data samples to compact representations and fitting the mixture model simultaneously leads to the representation collapse problem. Regularizing the distribution of data points over the clusters is the prevalent strategy to avoid this issue. While this is sufficient to prevent full representation collapse, we show that a partial prototype collapse problem still exists in the DINO family of methods, that leads to significant redundancies in the prototypes. Such prototype redundancies serve as shortcuts for the method to achieve a marginal latent class distribution that matches the prescribed prior. We show that by encouraging the model to use diverse prototypes, the partial prototype collapse can be mitigated. Effective utilization of the prototypes enables the methods to learn more fine-grained clusters, encouraging more informative representations. We demonstrate that this is especially beneficial when pre-training on a long-tailed fine-grained dataset.

1 Introduction

Self-supervised learning (SSL) is an effective approach to learn representations from unlabelled datasets. SSL methods have progressed rapidly in recent years and even surpassed the performance achieved by supervised training on several downstream tasks [10, 13, 22, 25, 47]. Broadly, SSL methods can be categorized into contrastive and non-contrastive methods. In contrastive methods [11, 24, 34], all data samples repel all other data samples resulting in an approximately uniform distribution of representations in the latent space [42]. Recent state-of-the-art SSL methods [12, 22, 25, 47] use Vision Transformers [17] and non-contrastive training methods. The prototypical formulations used in the DINO family of methods [10, 21, 31, 36, 47] enable data samples belonging to the same semantic cluster to concentrate while only repelling other clusters. Such methods learn representations that are effective at nearest neighbor tasks and few-shot learning.

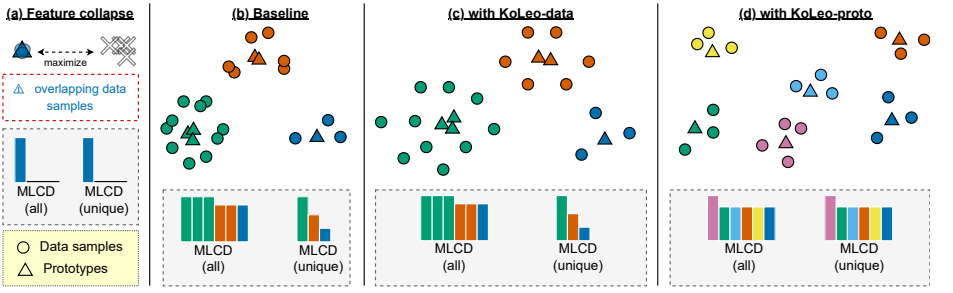

Figure 1: (a) The DINO family of methods result in a trivial full representation collapse without any regularization. (b) Using MLCD regularization such as centering and sharpening prevents full representation collapse but a partial prototype collapse still occurs. (c) KoLeo-data proposed in Oquab et al. [36] spreads the data representations further apart but does not address the partial prototype collapse. Note that the method (both baseline and with KoLeo-data) uses the partial prototype collapse to achieve a MLCD closer to a uniform distribution over all the prototypes. But the MLCD over only the unique prototypes is non-uniform. (d) We propose KoLeo-proto regularization that explicitly encourages diverse prototypes and prevents partial prototype collapse.

A common problem in this family of methods is the representation collapse. This originates from the simultaneous learning of the image representations as well as the clustering parameters. All existing methods regularize the marginal latent class distribution in order to prevent collapse. We show that these methods are still affected by a partial prototype collapse (i.e. some groups of prototypes converge to the same vector), resulting in much fewer unique prototypes compared to the initialized number (K). We consider a prototype to be unique if it is at least ϵ distance away from all other prototypes. We show an illustration of this in Figure 1. Moreover, varying the hyperparameter K has limited effect on the number of unique prototypes. The consequence is that the number of learned clusters cannot be reliably controlled through the hyperparameter. Hence, it is thus far unclear what impact varying the number of clusters will have on these methods.

Contributions: We formally define a partial prototype collapse and demonstrate its occurrence in the DINO family of methods, one of the most prominent family of SSL methods currently. We propose KoLeo-proto regularization to prevent such a collapse by explicitly encouraging diverse prototypes by maximizing their differential entropy. Then, we study the downstream impact of effective utilization of the prototypes. For datasets like Imagenet with uniform class distribution, we find this to be beneficial for few-shot learning (FSL) and marginally improves performance in full data scenarios. However, we observe a trade-off that exists between FSL performance on the pre-training dataset and transfer performance, that is consistent with other methods that report improved FSL performance. When pre-training on a long-tailed dataset such as iNaturalist-2018, we observe a clear performance gain when classifying the same dataset without affecting the transfer performance.

2 Background

The DINO-family of methods [2, 3, 10, 21, 31, 47] use the pretext task of assigning data to K latent classes with multi-view class consistency. Consider an encoder model that produces

a L^2-normalized representation $\boldsymbol{y} = g_{\boldsymbol{\theta}}(\boldsymbol{x})$ such that $\|\boldsymbol{y}\| = 1$, for a data point $\boldsymbol{x}$ using parameters $\boldsymbol{\theta}$. The probability of assigning a data point to a latent class k under the assumption of a latent class prior π_k is given by: $P_k(\boldsymbol{y}) = \Pr(z = k|\boldsymbol{y}) = \frac{\pi_k \Pr(\boldsymbol{y}|z=k)}{\sum_{j=1}^{K} \pi_k \Pr(\boldsymbol{y}|z=j)}$. With a uniform class prior $\pi_k \equiv 1/K$ (which is true in most prior work [2]), Govindarajan et al. [21] showed that the prototypical formulation in the DINO family corresponds to a von Mises-Fisher mixture model, with parameters $\{\boldsymbol{\mu}_k, \kappa_k\}$ and a normalization constant $C_p(\kappa_k)$

$$P_k(\boldsymbol{y}) = \frac{C_p(\kappa_k) \exp\langle \kappa_k \boldsymbol{\mu}_k, \boldsymbol{y} \rangle}{\sum_{j=1}^{K} C_p(\kappa_j) \exp\langle \kappa_j \boldsymbol{\mu}_j, \boldsymbol{y} \rangle}. \tag{1}$$

Here, $\boldsymbol{\mu}_k$ is the mean vector (a.k.a prototype) with $\|\boldsymbol{\mu}_k\| = 1$ and $\kappa_k > 0$ is the precision, which is a measure of concentration around the mean vector. The pre-training objective minimizes the KL-divergence between the latent class distributions of multiple views of each image. This task has a trivial solution where all data points can be mapped to the same representation. To prevent this collapse, it is essential to add some form of regularization to the training objective. The regularization techniques used in such methods can be motivated using the two requirements: (i) *the model should learn distinct clusters* and (ii) *spread the data over all these clusters*. The collapse where one or a few components dominate violates requirement-(ii). The collapse of individual probability distributions to uniform distributions implies that all the prototypes are equidistant from all the data representations. In practice, this leads to all prototypes collapsing to the same vector, which violates requirement-(i).

Connection between DINO and contrastive learning: Contrastive learning typically uses the normalized temperature-scaled cross entropy loss based on cosine similarities. Then, the probability distribution of a query representation $\boldsymbol{y}_q$ being similar to a set of candidate representations $\boldsymbol{y}_k$ is defined as: $P_k(\boldsymbol{y}_q) = \frac{\exp(\langle \boldsymbol{y}_q, \boldsymbol{y}_k \rangle / \tau)}{\sum_{j=1}^{K} \exp(\langle \boldsymbol{y}_q, \boldsymbol{y}_j \rangle / \tau)}$. SimCLR [10] uses candidate representations from the same batch and MoCo [24] uses a memory bank instead to avoid large batch sizes. Comparing this to Eq. (1), one can observe that the prototypes in DINO can be viewed as exemplary representatives of the dataset, replacing the memory bank. Thus, the DINO family of methods are a sparse variant of sample-contrastive methods [20].

3 Marginal latent class distribution

Before discussing a newly identified mode of collapse in the next section, we review and provide a unified understanding of some of the regularization techniques proposed in the literature to avoid collapse. We define the marginal latent class distribution (MLCD) as the probability vector with elements, $\bar{p}_k = \mathbb{E}_{\boldsymbol{x}}[P_k(g_{\boldsymbol{\theta}}(\boldsymbol{x}))]$. To our knowledge, all existing methods avoid representation collapse by regularizing the MLCD. Specifically, the MLCD is encouraged to match a prescribed prior distribution. A uniform prior is the default choice except for Assran et al. [3], who propose a power law distribution to better adapt the model to long-tailed data.

Adjusting the target distributions such that the MLCD matches a prior distribution can be posed as an entropy-regularized optimal transport problem, which can be solved using the Sinkhorn-Knopp (SK) algorithm [13]. SK is typically run for a few iterations and adds a small but noticeable computational overhead. Caron et al. [10] proposed centering, a simpler and computationally efficient method to adjust the target distributions. A key distinction between Sinkhorn-Knopp and centering is that they adjust the target distributions $P_k^{(t)}(\boldsymbol{y})$

based on batch and moving average estimates of the MLCD, respectively. On the other hand, Assran et al. [2, 3] add a prior-matching penalty on the batch-estimates of MLCD obtained from the online distributions $P_k^{(o)}(\boldsymbol{y})$. The penalty is defined as the KL-divergence between the MLCD and the prior distribution. With a uniform prior, this is equivalent to maximizing the entropy of MLCD, known as mean entropy maximization.

Is the centering adjustment ad-hoc? At first glance, the centering adjustment in DINO might appear somewhat ad-hoc. However, we find that probability centering (PC) [21] is closely connected to SK. Consider a batch of B logit scores over K latent classes $\boldsymbol{L} \in \mathbb{R}^{B\times K}$ and corresponding probability distributions $\boldsymbol{P}$. The SK adjusted (1 iteration) probability distributions are obtained as follows (derivation in A.2 of supplementary):

$$\tilde{\boldsymbol{P}}_{b,k}^{(\text{sk1})} = \frac{\exp(\boldsymbol{L}_{b,k} - \log(\frac{1}{B}\sum_b \boldsymbol{P}_{b,k}))}{\sum_{j=1}^{K}\exp(\boldsymbol{L}_{b,j} - \log(\frac{1}{B}\sum_b \boldsymbol{P}_{b,j}))}. \tag{2}$$

On the other hand, the probability centered distributions are obtained as follows, where the centering parameter c_k is calculated as a moving average estimate with momentum rate m:

$$\tilde{\boldsymbol{P}}_{b,k}^{(\text{pc})} = \frac{\exp(\boldsymbol{L}_{b,k} - c_k)}{\sum_{j=1}^{K}\exp(\boldsymbol{L}_{b,j} - c_j)}; \quad c_k \leftarrow mc_k + (1-m)\log\left[\frac{1}{B}\sum_{b=1}^{B}\boldsymbol{P}_{b,k}\right]. \tag{3}$$

Comparing Eq. (2) and Eq. (3), we observe that probability centering is equivalent to one iteration of SK with the key distinction that the logit adjustment is calculated as a moving average instead of a batch estimate. We compare them empirically in A.5.1 of supplementary.

4 Partial prototype collapse

Regularizing the MLCD enables the methods to meet the requirement of spreading data over clusters. However, the MLCD depends on both the data representations and the prototypes. Given a set of frozen data representations, a method can achieve MLCD matching a prior distribution simply by manipulating the prototypes (see Figure 1). This holds true for all existing methods in the DINO family, as they all regularize the MLCD in a similar manner, as discussed in the previous section. Sharpening prevents the extreme case when all prototypes collapse to the same vector. However, except for this limited guardrail, the existing regularization techniques do not ensure that the methods learn unique prototypes. We define the term *partial prototype collapse*, where only a significantly small proportion of the learned prototypes are unique.

Backbone	Method	Initialized prototypes (K)	Unique prototypes (M)
ViT-S/16	DINO-vMF	65536	1157
ViT-B/16	DINO-vMF	65536	939
ViT-B/16	iBOT	8192	875
ViT-B/16	iBOT-vMF	8192	1170
ViT-L/16	iBOT	8192	969
ViT-S/16	MSN*	8142	3363
ViT-S/16	PMSN*	8142	3005

Table 1: Number of unique prototypes in existing models with $\varepsilon = 0.025$ (default pre-training: ImageNet-1K, $*$: iNat-2018)

Definition 4.1 (Partial prototype collapse)**.** Consider the set $W = \{\boldsymbol{\mu}_k : k = 1,...,K\}$ of K prototype vectors, $\boldsymbol{\mu}_k$ such that $\|\boldsymbol{\mu}_k\| = 1$. A *partial prototype collapse* (of degree M and ε distance) is said to have occurred if there exists a set of M disjoint partitions of prototype vectors $V_m \subset W$, $m = 1,...,M$, and M representative prototype vectors $\boldsymbol{v}_m \in V_m$, such that for all $m = 1,...,M$, $1 - \boldsymbol{v}_m^\top \boldsymbol{\mu}_j < \varepsilon$, for all $\boldsymbol{\mu}_j \in V_m$. The set of M *unique prototypes* is defined as

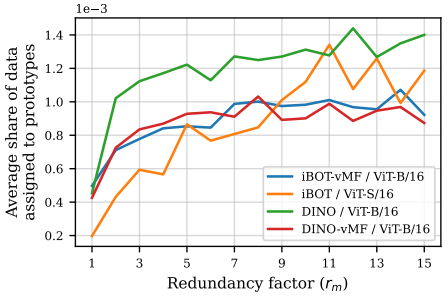

Figure 2: We reassign data to only the M unique representative prototypes and compute the average proportion of data assigned to prototypes having specific redundancy factors. We find that the models tend to assign a larger proportion of data to prototypes with higher redundancy factors. This holds true for DINO and iBOT with different backbones.

$U = \{\boldsymbol{v}_m\}_{m=1}^{M}$. For each representative prototype, the *redundancy factor* r_m is defined as the size of the corresponding set partition, $r_m = |V_m|$.

Investigating learned MLCD and prototypes: When training with MLCD regularization, the DINO family of methods are prone to partial prototype collapse since it enables the method to spread probability mass associated with each unique prototype across its ε-set of redundant prototypes. This acts as a shortcut to match the MLCD to the specified prior distribution. Govindarajan et al. [21] make an empirical observation that the DINO models used significantly smaller number of unique prototypes compared to the hyperparameter K. However, this problem is neither studied further nor addressed by their proposed method. Based on our definition of partial prototype collapse and using a cosine distance metric, we investigate the prototypes learned by several self-supervised clustering methods that use a prototypical formulation, from SwAV [9] to iBOT [47]. In Table 1 and Table A4 (in supplementary), we show that such a collapse exists in all the considered methods. We observe that prototypes with a higher redundancy factor tend to be assigned a larger proportion of the data samples (see Figure 2). Hence, the partial prototype collapse serves as a shortcut to achieve a MLCD closer to the specified uniform prior in these works. This shortcut is important to be aware of, if the intention is to encourage the MLCD to match a specific non-uniform distribution based on knowledge about the dataset domain. In addition, this means that the hyperparameter K does not play its intended role of controlling the number of clusters.

4.1 Regularizing prototype distribution

The number of latent classes is an important choice in clustering as this controls the fine-grainedness of the clusters. Firstly, this controls the difficulty of the self-supervision task. Secondly, more informative representations are required to discriminate between more fine-grained latent classes. With this motivation, we believe that the number of prototypes is an important design choice in SSL. However, prior works have found inconsistent results when ablating for this choice, likely because of the occurrence of partial prototype collapse. Given that we want the prototypes to be as diverse as possible, a meaningful choice is to encourage the prototypes $\boldsymbol{W} = \{\boldsymbol{\mu}\}_{k=1}^{K}$ to be uniformly distributed in the latent space. We propose to achieve this by maximizing the differential entropy of the prototype vectors, obtained using the Kozachenko-Leonenko estimator [5, 28, 39], $h_{\text{kl}}(\boldsymbol{W}) = -\frac{1}{K}\sum_{k=1}^{K}\log(d_k)$, where $d_k = \min_{i\neq k}\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_i\|$. We efficiently compute an estimate of $\mathcal{L}_{\text{KP}} = h_{\text{kl}}(\boldsymbol{W})$ by randomly partitioning the prototypes into batches and we show in A.3 of supplementary that this adds negligible computational overhead. We verify in section 6.1 that this regularization can mitigate the partial prototype collapse. Then, we focus on our main goal of studying the

downstream impact of effectively utilizing the initialized prototypes through various experiments that evaluate the learned representations.

5 Related Work

Connection to DINOv2: Our proposed KoLeo-proto regularization is formulated similar to Sablayrolles et al. [39]. Recently, DINOv2 [36] proposed the KoLeo-data regularization which uses a similar formulation but applied to spread the data representations instead of the prototypes. Hence, DINOv2 can be viewed as an interpolation between the uniformly distributed representations of contrastive learning and clustered representations of the DINO family. In contrast, KoLeo-proto preserves the clustered representations of DINO and encourages the method to learn diverse clusters. We illustrate this difference in Figure 1.

Regularizations in clustering-based SSL: We provide an extended discussion of clustering based SSL methods in A.1 of supplementary and focus our discussion on the regularization methods in this section. Asano et al. [1] and Caron et al. [2] used the Sinkhorn-Knopp (SK) algorithm to regularize the MLCD [15]. This is shown by Assran et al. [3] to encourage the MLCD to match a uniform prior. While SK requires multiple iterations for convergence, a simpler and computationally cheaper approach known as centering is proposed in DINO [10] and also used in EsViT [31] and iBOT [47]. Govindarajan et al. [21] proposed probability centering, that computed the centering parameter in the probability space instead of the logit space. MSN [2] and PMSN [3] proposed to add an explicit prior matching penalty to encourage the MLCD to align with prescribed prior distributions. Methods using the prior-matching penalty and SK depend on batch estimates of the MLCD. On the other hand, centering uses moving average estimates; we showed the connection of probability centering to SK in section 3. While all the above methods regularize the MLCD, we show the occurrence of a partial prototype collapse by investigating the prototypes learned by existing pre-trained models. We propose a new KoLeo-proto regularization as a tool to prevent this collapse and study the downstream impact of effectively utilizing the prototypes.

Pre-training on long-tail datasets: Most SSL methods are evaluated by pre-training on ImageNet with a uniform class distribution and there is limited research on pre-training SSL methods on long-tailed datasets. Caron et al. [8] investigated pre-training on a large uncurated dataset. Recently, Kukleva et al. [30] explored the benefits of using temperature schedules in the context of contrastive learning. Assran et al. [3] showed that pre-training on a long-tailed dataset can benefit from choosing an appropriate long-tail prior. We investigate the impact of effective prototype utilization when pre-training on a long-tailed dataset in section 6.3. Yang et al. [45] overcame a minority collapse issue [19] in supervised long-tailed classification with a fixed classification layer based on equiangular tight frames (ETF) geometry. However, this comes with the implicit assumption that all class prototypes should be equidistant which is a strong assumption for the latent classes learned in SSL.

6 Experiments

To study the MLCD and prototype regularizations, we focus on iBOT, which is a strong recent baseline among the DINO family of methods and also used as the foundation for DINOv2 [36]. We pre-train the models on the ImageNet-1K dataset [16] by modifying the public codebase of iBOT. We use the same hyperparameter settings as in iBOT for different

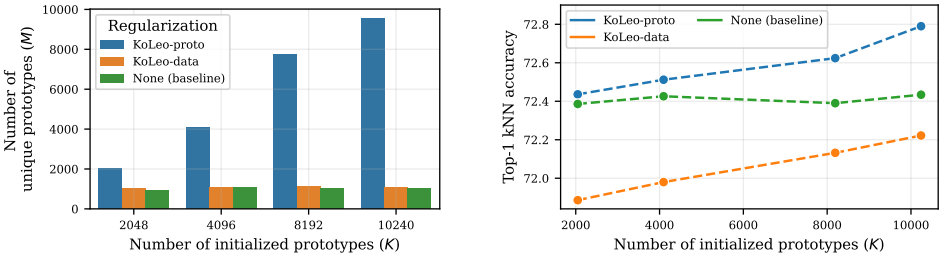

Figure 3: (left) The number of unique prototypes are similar for the baseline and KoLeo-data regularization at different number of initialized prototypes. With KoLeo-proto, most of the initialized prototypes remain unique. This means that the hyperparameter K can meaningfully control the number of learned clusters. (right) The number of initialized prototypes has no impact on the baseline performance. With any form of KoLeo-regularization, more prototypes lead to better performance and KoLeo-proto consistently performs best.

ViT backbones (see configuration details in A.4.1 and additional experiment with a CNN backbone in A.5.5 of supplementary) and use the vMF normalized variants [21], which are shown to produce stable trainings and improved performance. We found that probability centering consistently performs better than the other alternatives (Sinkhorn-Knopp and Mean Entropy Maximization) at different compute budgets. In all the following experiments, we use the vMF normalized iBOT with MLCD regularized using probability centering.

6.1 Prototype regularization

We add our proposed KoLeo-proto regularization to the iBOT-vMF baseline, resulting in the overall loss objective, $\mathcal{L} = \mathcal{L}_{\text{iBOT}} + \lambda \mathcal{L}_{\text{KP}}$. These results are indicated by "(kp)". Similarly, we indicate the KoLeo-data regularization used by Oquab et al. [36] as "(kd)". We use $\lambda = 0.1$ and observe that such a small λ is sufficient to mitigate partial prototype collapse (see ablation in A.5.3 of supplementary). In Figure 3, we compare the number of unique prototypes M when we vary the initialized number of prototypes hyperparameter K. With the baseline and KoLeo-data regularization, changing K has no impact on the number of unique prototypes learned by the method, which is significantly smaller than the initialized number of prototypes. This indicates the occurrence of partial prototype collapse. With KoLeo-proto regularization, we observe that $M \approx K$ and hence the hyperparameter K reliably controls the number of learned clusters.

We observe that the baseline shows similar performance at different numbers of initialized prototypes. On the other hand, with KoLeo-data, the performance is worse than the baseline but continues to improve as the number of prototypes are increased. KoLeo-data encourages the data to spread on the hypersphere. Hence, data is assigned to more diverse prototypes compared to the baseline in the initial training phase. We conjecture that this initial training dynamic benefits from having more prototypes, even if many of these prototypes eventually collapse to the same vector. We limit the maximum number of prototypes to 10240 due to computational limitations. Computing probability distributions for all the tokens over more dimensions adds a large computational overhead. However, the KoLeo regularization itself only adds a negligible computational overhead (cf. A.3 in supplementary). With KoLeo-proto, we observe around 0.1% improvement in accuracy when adding every

Method	kNN	Linear	Finetuning	1% data	5 img/cls	2 img/cls	1 img/cls	Avg. Transfer
ViT-Base/16								
DINO-vMF	77.4	78.8	83.6	70.4	66.1	59.3	50.3	86.8
MSN	73.3	74.8	–	69.1	65.5	58.9	49.8	85.3
WE-SSL	77.2	78.9	–	71.5	68.3	**62.4**	**53.7**	–
iBOT-vMF	78.7	80.3	**84.1**	72.3	68.3	61.1	51.6	**87.3**
iBOT-vMF (kp)	**78.8**	**80.5**	**84.1**	**72.7**	**69.1**	62.0	52.5	86.6
ViT-Small/16								
DINO-vMF	74.7	77.0	81.8	65.0	59.1	49.4	39.2	85.4
MSN	74.9	76.6	–	67.2	62.8	55.8	47.1	84.1
WE-SSL	75.2	77.4	–	**68.7**	**65.1**	**58.9**	**50.1**	84.6
iBOT-vMF	75.3	**77.9**	**82.3**	66.4	60.6	51.1	40.7	85.5
iBOT-vMF (kp)	**75.5**	**77.9**	**82.3**	67.0	61.1	51.7	41.6	**85.6**

Table 2: ImageNet classification with full data and few-shot scenarios and transfer learning.

2K additional prototypes. Overall, increasing the number of prototypes from 2K to 10K results in a 0.4% improvement. Further scaling of the number of prototypes can bring larger performance gains which should be feasible with the efficient implementation in DINOv2.

6.2 Pre-training with ImageNet

We pre-train iBOT-vMF with efficient prototype utilization using KoLeo-proto regularization for ViT-S/16 and ViT-B/16 backbones. To ensure fair comparison, we set the number of prototypes to 8192, similar to iBOT. Hence, any changes in performance can be associated to only the effective prototype utilization. In Table 2, we report the top-1 accuracies obtained using kNN and linear classification based on frozen backbone features, few-shot accuracies averaged over 3 different splits and the accuracy obtained after fine-tuning. For kNN, linear evaluation and finetuning, we follow the same protocol as in DINO and iBOT. We perform few-shot evaluation similar to Assran et al. [2] and use the provided data splits. We compare against the iBOT-vMF baseline, MSN and the best performing models from WE-SSL [38]. We observe on par or marginal improvements for kNN, linear and fine-tuned classification performance. The kNN performance improvement with respect to the baseline at 8192 prototypes after full-scale pre-training mirrors the improvement (+0.2%) observed after the small-scale ablation in Figure 3. This suggests that by using an even larger number of prototypes one can improve the performance further with efficient prototype utilization (cf. Figure 3), which we found to not be the case for the baseline in ablation experiments.

We find larger gains for few-shot learning (FSL) performance when adding KoLeo-proto to the baseline, even at 8192 prototypes. Note that the prediction head architecture and other hyperparameters are tuned in WE-SSL to achieve the best FSL performance with ViT-S/16. This explains the significantly better results achieved by WE-SSL with ViT-S/16. With ViT-B/16, iBOT-vMF (kp) outperforms WE-SSL at 1% and 5 img/cls settings. Note that iBOT-vMF can be tuned similar to WE-SSL but we have not investigated this. Instead, we focus on studying the impact of effective utilization of the prototypes on general downstream performance and do not perform any task-specific tuning.

Transfer learning: We conduct linear classification experiments on the standard suite of datasets trained using features extracted from a frozen pre-trained model. In Table 2, we report the accuracies averaged over all datasets. The detailed results and evaluation setup are provided in A.5.6 of supplementary. We observe on par or decreased transfer performance with effective prototype utilization compared to the iBOT-vMF baseline. Interestingly, we note that the transfer performance decreases also in other methods that improve

few-shot learning performance such as MSN [2] and WE-SSL [33] compared to their DINO baseline. This indicates that tuning for few-shot learning performance can potentially harm transfer performance. We hypothesize that certain features that improve few-shot learning performance on the pre-training dataset could be too specific to the pre-training data and do not generalize to other datasets considered for transfer learning. Compared to these methods, our proposed regularization leads to better transfer performance. There appears to be a trade-off between few-shot learning performance on the pre-training dataset and transfer learning performance. Currently, it is unclear why such a trade-off exists and this requires further investigation. Note that DINOv2 constructs the LVD142M pre-training dataset by finding images similar to the suite of transfer datasets of interest, thus limiting the domain gap between the pre-training and transfer datasets.

6.3 Pre-training with iNaturalist-2018

Most SSL methods are pre-trained on ImageNet which is well-curated and contains uniformly distributed data across its classes. It is of practical interest to pre-train SSL methods on data collected in the wild, which is often long-tailed. We study pre-training the DINO family of methods on long-tailed datasets, which has gained limited attention. We consider the iNaturalist-2018 (iNat18) dataset [1] which is around 1/3rd of the size of ImageNet and contains a long-tail distribution of data from 8142 classes. We pre-train all the models for 300 epochs using the default publicly available hyperparameters. For MSN and PMSN [2, 3], we choose the regularization strength λ based on a hyperparameter search (see A.4.2 in supplementary for details). This analysis of regularization strength λ indicated that weakly encouraging a uniform prior in MSN produces better performance than using a long-tailed prior as in PMSN. With this motivation, we retain the uniform prior assumption of the other methods. In Table 3, we report the top-1 classification accuracy obtained using a linear and a fine-tuned classifier. For linear classification, we follow a similar protocol as in the ImageNet experiments. For fine-tuning, we use longer trainings with a smaller learning rate as in DINO [10] (see details in A.4.5 of supplementary). We consider iBOT-vMF as our baseline method, which significantly outperforms MSN and PMSN.

For ViT-S model, we find that both KoLeo regularization methods bring performance benefits compared to the baseline. After evaluating the two forms of KoLeo regularization on the ViT-B model as well, we conclude that KoLeo-proto regularization performs best. With partial prototype collapse, models learn more coarse-grained latent classes where the number of unique prototypes are less than the number of classes (see M in Table 3). Then, the learned clusters are likely to have merged several of the fine-grained classes. This is mitigated when the prototypes are effectively utilized, leading to more diverse clusters and hence, more informative representations which are beneficial for long-tailed and fine-grained classifi-

Method	M	Linear	Fine-tuned	Avg. Transfer
ViT-Small/16				
DINO-vMF	1380	49.7	68.5	71.7
iBOT-vMF	1804	50.1	**69.4**	<u>71.8</u>
iBOT-vMF (kd)	1843	<u>50.5</u>	69.1	71.5
iBOT-vMF (kp)	7895	**51.1**	<u>69.3</u>	**72.0**
MSN ($\lambda = 1$)	3363	43.8	63.5	67.0
PMSN ($\lambda = 5$)	3005	41.8	64.2	66.1
ViT-Base/16				
iBOT-vMF (kd)	1634	50.4	73.3	73.2
iBOT-vMF (kp)	7573	**51.4**	**74.0**	**73.7**

Table 3: iNat-2018 (linear probing and fine-tuning) and avg. transfer classification accuracies

[1] This dataset was used for academic purposes only.

cation. Hence, iNat-2018 pre-training benefits more from effectively utilizing prototypes compared to the Imagenet experiments. We report average transfer learning accuracies in Table 3 and detailed results in Table A9 of supplementary. In contrast to the ImageNet experiments, effective utilization of prototypes in KoLeo-proto is also beneficial for transfer performance.

7 Conclusion

We identified the occurrence of a previously unnoticed mode of collapse in the DINO family of methods, termed as *partial prototype collapse* that results in significant redundancies in the prototypes. As a consequence, the hyperparameter controlling the number of prototypes did not perform its intended role of controlling the number of clusters learned by the model. We proposed the KoLeo-proto regularization to encourage the model to learn diverse prototypes. By adding our proposed regularization, we showed that the initialized prototypes are effectively utilized. With effective prototype utilization, scaling the number of prototypes is useful in learning better image representations of the underlying dataset. Using the same moderate number of 8K prototypes as before, we showed that few-shot learning performance can be improved and full data trainings can be marginally improved. As indicated in our ablation experiments, it seems possible that further scaling the number of prototypes can result in more significant improvements. However, we observed a worse transfer performance and this trade-off is consistent with other methods that specifically improve few-shot learning. On the other hand, we found that learning fine-grained clusters on a long-tailed fine-grained dataset such as iNat-2018 is more beneficial, indicated by the larger performance gains achieved using a similar number of prototypes.

We have shown that the hyperparameter for the number of prototypes can be reliably controlled using our regularization. This has broad implications on applying methods from the DINO family. One can better understand the impact of using different numbers of clusters in the self-supervised pretext task for their own dataset and method of choice. This could vary depending on the domain of the dataset and how fine-grained the semantic concepts are in that domain. Computing probability distributions over a large number of latent classes comes at a significant computational cost (see A.5.4 in supplementary). If indeed a small number of clusters are sufficient for some dataset, effectively utilizing fewer prototypes can help in reducing computational expenses.

Acknowledgments

This research is financially supported by the Swedish Research Council via the project *Handling Uncertainty in Machine Learning Systems* (contract number: 2020-04122), the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and the Excellence Center at Linköping–Lund in Information Technology (ELLIIT). The computations were enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

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] Mahmoud Assran, Randall Balestriero, Quentin Duval, Florian Bordes, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, and Nicolas Ballas. The hidden uniform cluster prior in self-supervised learning. In *ICLR*, 2023.

[4] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *ICLR*, 2022.

[5] Jan Beirlant, Edward J Dudewicz, László Györfi, Edward C Van der Meulen, et al. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 1997.

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.

[7] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[8] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019.

[9] 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.

[10] 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.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

[13] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

[14] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[15] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013.

[16] 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.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[18] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *CVPR*, 2021.

[19] Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proceedings of the National Academy of Sciences*, 2021.

[20] Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann LeCun. On the duality between contrastive and non-contrastive self-supervised learning. In *ICLR*, 2023.

[21] Hariprasath Govindarajan, Per Sidén, Jacob Roll, and Fredrik Lindsten. DINO as a von mises-fisher mixture model. In *ICLR*, 2023.

[22] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *NeurIPS*, 2020.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[25] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[26] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[27] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021.

[28] Lyudmyla F Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 1987.

[29] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[30] Anna Kukleva, Moritz Böhle, Bernt Schiele, Hilde Kuehne, and Christian Rupprecht. Temperature schedules for self-supervised contrastive methods on long-tail data. In *ICLR*, 2023.

[31] 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.

[32] Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101, 2022. URL https://data.caltech.edu/records/20086.

[33] 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.

[34] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

[35] 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.

[36] 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.

[37] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012.

[38] 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.

[39] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Spreading vectors for similarity search. In *ICLR*, 2019.

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

[41] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[42] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020.

[43] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 2021.

[44] 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.

[45] Yibo Yang, Shixiang Chen, Xiangtai Li, Liang Xie, Zhouchen Lin, and Dacheng Tao. Inducing neural collapse in imbalanced learning: Do we really need a learnable classifier at the end of deep neural network? *NeurIPS*, 2022.

[46] 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.

[47] 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.