Improving the Generalization of Supervised Models

Mert Bulent Sariyildiz, Yannis Kalantidis, Karteek Alahari, Diane Larlus

Improving the Generalization of Supervised Models

Mert Bulent Sariyildiz[1,2] **Yannis Kalantidis**[1] **Karteek Alahari**[2] **Diane Larlus**[1]

[1] NAVER LABS Europe [2] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK

Abstract

We consider the problem of training a deep neural network on a given classification task, e.g., ImageNet-1K (IN1K), so that it excels at that task as well as at other (future) transfer tasks. These two seemingly contradictory properties impose a trade-off between improving the model's generalization while maintaining its performance on the original task. Models trained with self-supervised learning (SSL) tend to generalize better than their supervised counterparts for transfer learning; yet, they still lag behind supervised models on IN1K. In this paper, we propose a supervised learning setup that leverages the best of both worlds. We enrich the common supervised training framework using two key components of recent SSL models: multi-scale crops for data augmentation and the use of an expendable projector head. We replace the last layer of class weights with class *prototypes* computed on the fly using a memory bank. We show that these three improvements lead to a more favorable trade-off between the IN1K training task and 13 transfer tasks. Over all the explored configurations, we single out two models: **t-ReX** that achieves a new state of the art for transfer learning and outperforms top methods such as DINO and PAWS on IN1K, and **t-ReX*** that matches the highly optimized RSB-A1 model on IN1K while performing better on transfer tasks. Project page and pretrained models: https://europe.naverlabs.com/t-rex

1 Introduction

Deep convolutional neural networks trained on large annotated sets of images like ImageNet-1K (IN1K) [12, 47] have shown strong generalization properties. This motivated their application to a broad range of transfer tasks including the recognition of concepts that are not encountered during training [13, 51].

Recently, models trained in a self-supervised learning (SSL) framework have become popular due to their ability to learn without manual annotations, as well as their capacity to surpass supervised models in the context of transferable visual representations. SSL models like MoCo [20], SwAV [6], BYOL [19], Barlow Twins [65] or DINO [7] do not leverage manually provided labels and yet exhibit stronger transfer learning performance than models [60] trained on the same dataset with annotations [50].

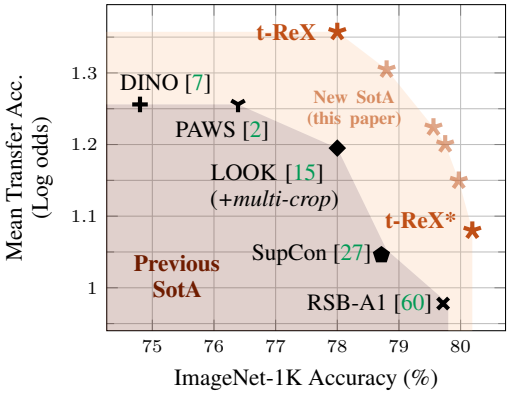


Figure 1: We present **t-ReX** and **t-ReX***, two ResNet50 models trained with an improved supervised learning setup on ImageNet-1K, with strong performance on both transfer learning (y-axis, averaged over 13 tasks) and IN1K (x-axis), respectively outperforming DINO [7] and RSB-A1 [60], the state of the art (SotA) along each dimension.

Preprint.

This achievement is on the one hand exciting, as SSL approaches do not require an expensive and error-prone annotation process, but also seemingly counter-intuitive as it suggests that access to additional information, i.e., image labels, actually hinders the generalization properties of a model. Models learned via SSL are however not able to match their supervised counterparts on IN1K classification, i.e., on the concepts seen during training. Top-performing SSL and semi-supervised methods like DINO [7] or PAWS [2] still result in 3-5% lower top-1 accuracy compared to optimized supervised models such as RSB-A1 [60].

In this paper, we argue that access to more information (in the form of manual annotations) should not hurt generalization, and we seek to improve the transferability of encoders learned in a supervised manner, while retaining their state-of-the-art performance on the supervised training task. The mismatch observed between IN1K and transfer performance [30, 50] suggests that this goal is not trivial. It has been shown, for example, that some popular regularization techniques like Label Smoothing [56], Dropout [54] or CutMix [64] that improve IN1K performance, actually lead to less transferable representations [30, 50].

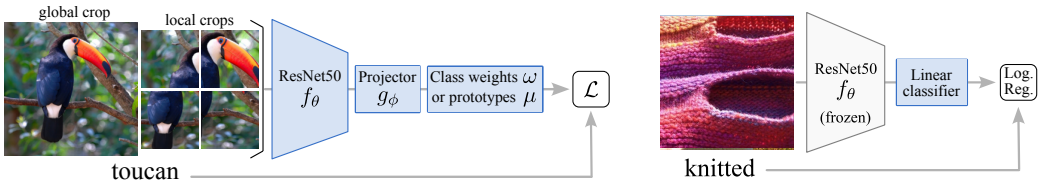
We identify two key training components from the most successful SSL approaches [7, 8, 19] that may lead to more transferable representations: multi-crop data augmentation [6] and the use of expendable projector heads [8]. More precisely, we first study the impact of image crops at multiple scales in the commonly used supervised learning setting with a cross-entropy loss, and observe that *multiple local crops* bring consistent gains on both IN1K and the transfer tasks. We also explore the use of *projectors*, i.e., auxiliary modules added after the encoder during training and discarded at test time. We observe these two improvements to significantly boost transfer learning performance. Finally, inspired by recent work [15, 27], we explore alternative training objectives and in particular we formulate a memory-based version of the nearest class means classifiers [38] that improves performance further by replacing the class weights usually learned under the common supervised learning setup with class *prototypes*, computed on-the-fly, via averaging features from a memory bank.

These observations motivate the design of a supervised learning setup that produces highly competitive models effective for *both* IN1K and transfer learning. We single out the two ResNet50 instantiations that perform best at one of these two dimensions, denoted as **t-ReX** and **t-ReX***. **t-ReX** exceeds the state-of-the-art transfer learning performance of DINO [7] or PAWS [2] and still performs much better than these two on IN1K classification. **t-ReX*** outperforms the state-of-the-art results of RSB-A1 [60] on IN1K while generalizing better to transfer tasks. We visualize the performance of these two selected models, together with those of other top-performing configurations from our setup in Figure 1, and compare it to state-of-the-art supervised, semi-supervised and self-supervised learning methods, across two dimensions: IN1K accuracy and mean transfer accuracy across 13 transfer tasks. This intuitively conveys how the proposed training setup *pushes the envelope* of the training-versus-transfer performance trade-off (from the "Previous SotA" region, to the "New SotA" one) and offers strong pretrained encoders that future approaches could build on.

Contributions. To summarize, our contribution is threefold. First, we propose a supervised training setup that produces competitive models for both IN1K and diverse transfer learning tasks, thanks to multi-crop data augmentation and an expendable projector added during training. Second, we reformulate the training objective to use online variants of the neighborhood component analysis [17] and the nearest class mean classifier [38]. We show that they can successfully be part of our training setup and that the latter can improve results further. Third, we thoroughly ablate training components as well as our training setup and loss functions. We present a set of models which all display a favorable trade-off along both dimensions: performance on the training task of ImageNet-1K and on the different transfer tasks. We single out, **t-ReX** and **t-ReX***, the models that respectively achieve the new state of the art on each dimension. We believe they can serve as strong baselines for the future.

2 Related work

Soon after the remarkable performance of AlexNet [33] on IN1K, the computer vision community started leveraging the fact that representations produced by deep networks trained for IN1K classification transfer to other datasets and tasks [13, 51]. Several works since then have proposed practical transfer approaches [18, 41, 66], while others have contributed to a formal understanding of those beneficial generalization properties [25, 31, 57, 63]. Recent work in this context [30, 50] has shown that the best representations for IN1K were not necessarily the ones transferring best. For instance,



(a) Supervised learning using multi-crop and a projector. (b) Transfer learning with a frozen model.

Figure 2: **Our proposed supervised learning setup** borrows multi-crop [6] and projectors [8] from SSL to train on IN1K (*left*). The projector g is discarded after training, and the ResNet backbone f is used as a feature extractor in combination with a linear classifier trained for each task, e.g., for texture classification on DTD [11] (*right*).

some regularization techniques or loss functions improving IN1K classification lead to underwhelming transfer results [30, 50]. A parallel line of work based on self-supervised learning [6, 8, 19, 20] focused on training models without manual labels, and demonstrated their strong generalization capabilities to many transfer datasets, clearly surpassing their supervised counterparts [50]. Yet, as expected, SSL models are no match to the supervised models on the IN1K classification task itself.

A few approaches have tackled the task of training supervised models that also transfer well and share motivation with our work. SupCon [27] extends SimCLR [8] using image labels to build positive pairs. As such, its formulation is close to neighborhood component analysis (NCA) [17]. It circumvents the need for large batches by adding a momentum and a memory similar to MoCo [20]. Supervised-MoCo [69] filters out false negatives in the memory bank of MoCo using image labels, while LOOK [15] modifies the NCA objective to only consider the closest neighbors of each query image. We experimentally observe that our model design leads to better transfer than all these works.

Our work is also related to the metric learning literature. Metric learning models are trained with labels but evaluated on unseen concepts, and often focus on specific and narrow class domains such as bird species or car models [45, 53, 70]. Therefore, they are not directly applicable to our problem. Moreover, their formulation usually involves computing pairwise image similarities [27, 39, 61], which does not scale to IN1K. Also related, adversarially robust classifiers have been shown to transfer well [48], but the level of noise applied during training needs to be carefully set for each task.

Departing from all these methods, we propose an effective training setup, which leverages multi-crop augmentation [6] and an expendable projector head [8], two key components in many successful SSL approaches [7, 9, 10, 19]. Multi-crop is typically used for creating diverse and challenging views of an image, for which the model is encouraged to learn consistent representations [2, 7]. A concurrent work [58] argues that multi-crop increases representation variance and should be used for online self-distillation, while it has recently been shown to also improve vision and language pretraining [28]. We show that this design works out-of-the-box also for supervised training on IN1K.

The use of expendable projectors comes from the observation that the last layer of a model is not necessarily the one producing features that transfer best. Early works on SSL evaluate representations from multiple layers throughout the model [5, 16, 18, 29, 68], as the last layer tends to overfit to the proxy task, e.g., clustering [5] or rotation prediction [16]. To train more generic features, SimCLR [8] adds a MLP projector after the encoder, that is discarded later. This design has become standard practice in SSL [7, 9, 19, 65]. Projectors are also used in recent supervised models such as SupCon [27] and LOOK [15], but none of these works studies the impact of the projector design choices on representation quality. Concurrent work [59] investigates the role of a projector for supervised pretraining and attributes its success on redundancy reduction across features. We show that gains from projectors are complementary to multi-crop as well as alternative training objectives.

3 An improved training setup for supervised learning

We present an improved supervised training setup for learning models that achieve high performance on both IN1K classification and a set of diverse transfer tasks. First, we present a standard supervised training method, which forms the basis for our setup (Section 3.1). We then enhance it with

multi-crop augmentations as well as an expendable projector, leading to our improved supervised learning approach (Section 3.2). We finally explore alternative training setups with the same core improvements, varying the training objective (Section 3.3). These different contributions lead to several successful models including the **t-ReX** and **t-ReX*** models that we have singled out.

After training each of these models, we perform transfer learning on several tasks by freezing the models' parameters, i.e., treating them as feature extractors, and combining them with a linear classifier for each transfer task (see Section 4). The transfer process is depicted in Figure 2b.

3.1 Background: Supervised learning on ImageNet-1K

Our setup trains a model (or *encoder*) f_θ, parameterized by θ. This model encodes an image I of size $H \times W \times 3$ into a transferable representation $x \in \mathbb{R}^d$. We follow the common protocol [15, 27, 30] and train all models on IN1K using a ResNet50 [21] backbone. Our choice of backbone is influenced by recent observations [60] that carefully optimized ResNet50 models perform on par with the best Visual Transformers (ViTs) [3] of comparable size on IN1K.

In the most common setup, $C = 1000$ class weights and $W = \{\omega_c\}_{c=1}^C$ are learned jointly with the encoder parameters θ, using a softmax cross-entropy loss. Here, we use the *cosine softmax* variant that first ℓ_2-normalizes both the representations x and the class weights W, as it was shown to improve IN1K performance [30]. Formally, let $y \in \{0,1\}^C$ be the C-dim one-hot label vector corresponding to feature $x = f_\theta(I)$, and let $\omega_c \in \mathbb{R}^d$ be the class weight for class c. The cosine softmax loss for image I is:

$$\mathcal{L}_{\text{CE}}^{\cos} = -\sum_{c=1}^{C} y_{[c]} \log \frac{\exp(\bar{x}^\top \bar{\omega}_c/\tau)}{\sum_{k=1}^{C} \exp(\bar{x}^\top \bar{\omega}_k/\tau)}, \text{ with } \bar{x} = x/\|x\| \text{ and } \bar{\omega}_c = \omega_c/\|\omega_c\|, \tag{1}$$

where $y_{[c]}$ denotes the c-th element of vector y, $\|\cdot\|$ is ℓ_2-normalization, and τ is a temperature hyper-parameter. A Stochastic Gradient Descent (SGD) optimizer with momentum is typically used, together with a cosine learning rate schedule [35, 52] that has been shown to generally improve performance.

Another important factor for learning effectively is data augmentation. The most common set of data augmentations goes back to GoogleNet [55] and consists of random horizontal flips and random resized crops of 224×224 pixels. This basic set is still one of the most popular [10, 15, 27], and similar to DINO [7], we follow this strategy, together with extreme color jittering. Optimized variants such as RSB-A1 [60] add augmentations like Mixup [67] or CutMix [64] that improve IN1K performance, but recent studies have found that they hurt transfer learning [50].

3.2 Improved supervised training setup

We start from the baseline described in Section 3.1, and enrich this standard supervised learning setup with the following components.

Multi-crop data augmentation. Recently, Caron *et al.* [6] introduced the use of many image crops of multiple scales and different model input sizes when learning invariance to data augmentation in the context of SSL. Their data augmentation pipeline, termed *multi-crop*, is defined over two sets of *global* and *local* crops that respectively retain larger and smaller portions of an image and are used in a distillation formulation. We adapt this component to our supervised setup.

Given an input image, we define two scale parameters, for global and local crops. Global crops follow the data augmentation protocol described in Section 3.1, while local crops are resized to 96×96 instead. Similar to [6, 7], we extract multiple global and local crops, respectively M_g and M_l. Figure 2a illustrates one global $M_g = 1$ and four local $M_l = 4$ crops. In Section 4, we explore the use of multi-crop for supervised learning, and study the effect of different hyper-parameters under that setting.

Expendable projector head. To countervail the lack of annotations, SSL approaches tackle proxy tasks, such as learning augmentation invariance. In order to prevent the encoder to build representations that overfit to a potentially unimportant pretext task, SSL architectures generally introduce an expendable projector head between the encoder and the loss function. On the contrary, for supervised learning, performance on the training task is a major goal in its own right. Here, we aim at learning supervised models that perform well on the training *and* on transfer tasks. These two requirements are not aligned and it is necessary to find a trade-off [30].

We argue that one can control this trade-off using projector heads in the context of supervised learning. Similar to SSL methods [7, 8, 9, 10], we introduce a Multi Layer Perceptron (MLP) projector as part of our supervised training pipeline. Let $g_\phi : \mathbb{R}^d \to \mathbb{R}^{d_b}$ denote the projector, parameterized by ϕ. g_ϕ is composed of an MLP with L hidden layers followed by a linear projection to a bottleneck of d_b dimensions. Each hidden layer is composed of a sequence of a linear fully-connected layer, batch-normalization [26] and a GeLU [23] non-linearity. We further apply ℓ_2-normalization to the output of g_ϕ and optionally also to the input. We illustrate this architecture in Figure 3. In our experiments (Section 4), we investigate how the number and dimension of hidden layers among other design choices affect the performance of the learned models. We observe that the design of this projector allows to control the desired trade-off between the performance on the training task and transferability.

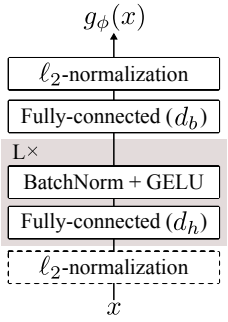


Figure 3: Architecture of the projector g_ϕ.

Training with cosine softmax cross-entropy loss. The complete training pipeline, using the components described so far, is illustrated in Figure 2a. It uses multi-crop data augmentation on each input image I and produces $M = M_g + M_l$ crops I_j, $j = 1, \ldots, M$. Each crop is individually input to the network composed of the encoder followed by the projector, and produces representation $x_j = f_\theta(I_j)$. We adapt the cosine softmax cross-entropy loss defined for image I in Equation (1), such that we average the loss over all (local and global) crops of an image:

$$\mathcal{L}^*_{\text{CE}} = -\frac{1}{M}\sum_{j=1}^{M}\sum_{c=1}^{C} y_{[c]} \log \frac{\exp(g_\phi(x_j)^\top \bar{\omega}_c/\tau)}{\sum_{k=1}^{C}\exp(g_\phi(x_j)^\top \bar{\omega}_k/\tau)}. \tag{2}$$

Similar to Equation (1), the class weights and the outputs of the projector are ℓ_2-normalized. All the parameters (i.e., encoder, projector head, and class weights) are trained with SGD. The dimension of the class weights is set according to the bottleneck dimension d_b of the projector head.

3.3 Improving supervised learning using memory-based class means

The previous section presents an improved training setup for the dominant supervised learning paradigm, i.e., using softmax cross-entropy over learned class weights. We now revisit two older supervised learning approaches, Neighborhood Component Analysis (NCA) [17] and Nearest Class Means (NCM) [38], so they could be adapted for large-scale training on IN1K and propose online, memory-efficient versions of those. In both cases we start from the improved setup described above and we use multi-crop augmentation and an expendable projector head discarded after training.

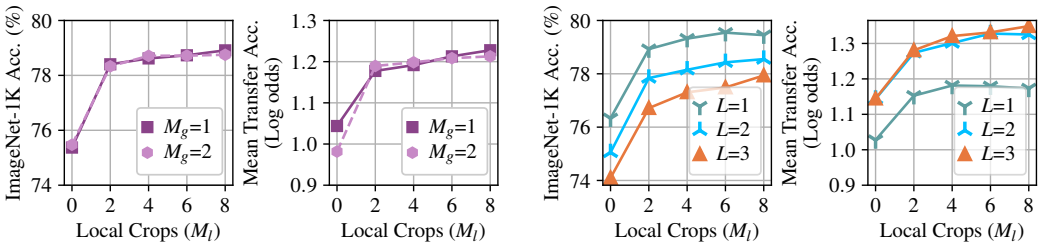
Online Component Analysis. Recent supervised learning methods like SupCon [27] or LOOK [15] are variants of the soft k-NN loss introduced in Neighborhood Component Analysis (NCA) [17]. We therefore explore a variant of our training setup where the loss directly minimizes the log NCA probabilities. Specifically, similar to the idea considered in MoCo [20], we use a memory bank $\mathcal{Q}$ which stores ℓ_2-normalized representations z output by the projector instead of using the full dataset to compute the soft k-NN, as the latter would be intractable. Our training setup (with multi-crop and a projector), then optimizes the following loss:

$$\mathcal{L}^*_{\text{OCA}} = -\frac{1}{M}\sum_{j=1}^{M}\sum_{c=1}^{C} y_{[c]} \log \frac{\sum_{z_c \in \mathcal{Q}_c}\exp(g_\phi(x_j)^\top z_c/\tau)}{\sum_{z_n \in \mathcal{Q}}\exp(g_\phi(x_j)^\top z_n/\tau)}. \tag{3}$$

where C is the number of classes, $\mathcal{Q}_c$ denotes samples in the memory bank that belong to class c and M is the number of image crops.

Since this is an "online" variant of the NCA objective, we refer to this training objective as Online Component Analysis or OCA. We experimentally show that this variant is on par with the one described in Section 3.2 on the training vs transfer trade-off. Yet the variant we propose next, an online version of NCM, leads to even better results.

Online Class Means. Inspired by the seminal Nearest Class Means (NCM) approach of Mensink *et al.* [38], we explore a variant where we define ω_c to be the *class mean* for each class c, i.e., the mean of the representations from that class. Given that we jointly learn class means and the representations,



(a) multi-crop without using a projector

(b) multi-crop using a projector

Figure 4: **Impact of the number of local crops** on IN1K performance (Top-1 Accuracy) and mean transfer Accuracy (log odds) over the 13 transfer datasets. *Left:* Varying the number of global crops without using a projector head. *Right:* Varying the number of hidden layers in the projector head.

computing exact class means at each iteration is computationally prohibitive. Instead, we formulate an *online* version of NCM that uses a memory bank.

Specifically, we store images in a memory following the process described in MoCo [20], and we compute a mean vector per class using only images stored in this memory bank. Formally, given the memory $\mathcal{Q}$, we do not learn class weights, but instead compute a *prototype* for each class that is computed on-the-fly as the average of the representations of images in memory that belong to that class. With $\mathcal{Q}_c$ denoting samples in the memory that belong to class c and $N_c = |\mathcal{Q}_c|$, the loss function becomes:

$$\mathcal{L}^*_{\text{OCM}} = -\frac{1}{M}\sum_{j=1}^{M}\sum_{c=1}^{C} y_{[c]} \log \frac{\exp(g_\phi(x_j)^\top \bar{\mu}_c/\tau)}{\sum_{k=1}^{C}\exp(g_\phi(x_j)^\top \bar{\mu}_k/\tau)}, \tag{4}$$

where $\bar{\mu}_c = \mu_c/\|\mu_c\|$, $\mu_c = {}^1/_{N_c}\sum_{z\in\mathcal{Q}_c} z$, and C and M are still the number of classes and crops per image.

Since we compute class prototypes in an "online" manner with the help of the memory queue, we refer to this training objective as Online Class Means or OCM.

Note that, to make sure the representations stored in the memory remain relevant as the encoder is updated during training, for both the OCA and OCM approaches, we follow MoCo [20] and store in memory representations from an exponential moving average (EMA) model trailing f_θ and g_ϕ. More details about the EMA model and resulting architectures can be found in Appendix A.2.

4 Experiments

We present a thorough evaluation of the different components and supervised training variants introduced in the previous section.

Protocol. All models are trained on the training set of ImageNet-1K (IN1K) [47]. Due to the computational cost of training models on IN1K, each configuration is trained only once. Given a model trained on IN1K, we discard all training-specific modules (e.g., the projector g_ϕ and class weights W), and use the encoder f_θ as a feature extractor, similar to [31, 50]. For each dataset we evaluate on, we learn a linear logistic regression classifier on top of pre-extracted features and independently optimize each classifier hyper-parameters *for every model and every evaluation dataset* using Scikit-learn [44] and Optuna [1] (details in Appendix B). We repeat this process 5 times with different random seeds and report the average accuracy (variance is negligible). Note that the feature extractor is never fine-tuned, and, because we start from pre-extracted features, no additional data augmentation is used when learning the linear classifiers.[1] This protocol is illustrated in Figure 2b.

Evaluation datasets and measures. We measure performance on the training task by evaluating classification accuracy on the IN1K validation set as well as on IN1K-v2 [46]. To evaluate transfer

[1] Although methods like [7, 31, 66] train linear classifiers with data augmentation or fine-tune the encoder while training classifiers, we found that such a protocol makes a proper hyper-parameter validation computationally prohibitive. We instead follow the linear evaluation protocol from [31, 50].

Table 1: **Effect of the projector's design.** We start from a default configuration, i.e., $L = 1$, $d_h = 2048$, $d_b = 256$ and with ℓ_2 normalization of the input (corresponding to highlighted rows in the tables), and ablate each hyper-parameter separately, training models with Equation (2). Unless otherwise stated, we use multi-crop with $M_g = 1$ and $M_l = 8$ (abbreviated below as "mc").

(a) # of hidden layers (L)

	IN1K	Transfer
no mc	76.0	1.06
mc	78.9	1.23
$L = 1$	79.8 ↑	1.15
$L = 2$	78.6	1.31
$L = 3$	77.5	1.33 ↓

(b) Hidden layer dim.

d_h	IN1K	Transfer
512	80.0 ↑	0.82
1024	80.0	1.06
2048	79.8	1.15
4096	79.8	1.20
8192	79.4	1.22 ↓

(c) Bottleneck dim.

d_b	IN1K	Transfer
128	79.8	1.14
256	79.8	1.15
512	79.9	1.16
1024	79.6	1.15
2048	80.0	1.15

(d) Input ℓ_2-norm

L	ℓ_2	IN1K	Transfer
1	–	79.4	1.17
1	✓	79.8	1.15
2	–	78.6	1.33
2	✓	78.6	1.31
3	–	77.9	1.35
3	✓	77.5	1.33

learning, we measure classification performance on 13 datasets: the 5 ImageNet-CoG [50] datasets that were recently proposed to measure concept generalization, plus 8 commonly used smaller-scale datasets: Aircraft [36], Cars196 [32], DTD [11], EuroSAT [22], Flowers [40], Pets [42], Food101 [4] and SUN397 [62] (see Table 5 in Appendix for more details on the datasets). We report two metrics: Top-1 accuracy on IN1K and transfer accuracy via log-odds [31] (see also Appendix B.3 for the exact formulation) averaged over the 13 transfer datasets. We also report individual results per dataset in Table 8 of Appendix.

Implementation details. f_θ is a ResNet50 [21] encoder, trained for 100 epochs with mixed precision in PyTorch [43] using 4 GPUs where batch norm layers are synchronized. We use an SGD optimizer with 0.9 momentum, a batch size of 256, 1e-4 weight decay and a learning rate of $0.1 \times {}^{\text{batch size}}/_{256}$ which is linearly increased during the first 10 epochs and then decayed with a cosine schedule. We set $\tau = 0.1$ and unless otherwise stated we use the data augmentation pipeline from DINO [7] with 1 global and 8 local crops (M_g = 1 and M_l = 8). The respective resolution of global and local crops is 224 and 96. Training one of our models takes up to 3 days with 4 V100 GPUs depending on its projector configuration.

4.1 Multi-crop augmentation and projector heads for supervised and transfer learning

Multi-crop data augmentation. We first evaluate multi-crop under the common supervised learning setup presented in Section 3.1, i.e., without the use of a projector. We train supervised models and study the effect of the number of global and local crops on IN1K and on transfer learning performance. In Figure 4a we report results using 1 or 2 global and 2, 4, 6 or 8 local crops, setting the scale parameters to be identical to the ones that [7] uses for ViTs, as we observed them to be the ones working best for ResNet50 as well (see Table 9 in Appendix for an ablation).

Results from our study brings the following three observations: a) training with local crops improves the performance on both IN1K and transfer tasks, b) there is no benefit in using 2 global crops instead of 1, c) although increasing the number of local crops generally helps, performance saturates with 8 local crops. *We set $M_g = 1$ and $M_l = 8$ for all subsequent evaluations.*

Note that using local crops comes at the expense of an increased training time; e.g., using 8 local crop doubles training time. We therefore conducted an experiment to see if longer training could have a similar effect and trained a model for 800 epochs using a single crop sampled from a wide scale range, i.e., able to focus on both big and small parts of the image. Unlike multi-crop this did not bring any gain (see Table 9 in Appendix for details).

Expendable projector head. We now study the impact of different architectural choices and hyper-parameters for the projector. We vary the number of hidden layers (L), the dimension of the hidden (d_h) and bottleneck (d_b) layers, and whether or not to ℓ_2-normalize the projector input (ℓ_2). We start from a default configuration: $L = 1$, $d_h = 2048$, $d_b = 256$ and with ℓ_2 normalized inputs. We ablate each parameter separately by training models optimizing Equation (2). We use multi-crop in all cases.

Results from this analysis are presented in Table 1. Table 1a shows that *the number of hidden layers (L) is an important hyper-parameter that controls the trade-off between IN1K and transfer performance*. Adding a projector head with a single hidden layer already improves the strong IN1K

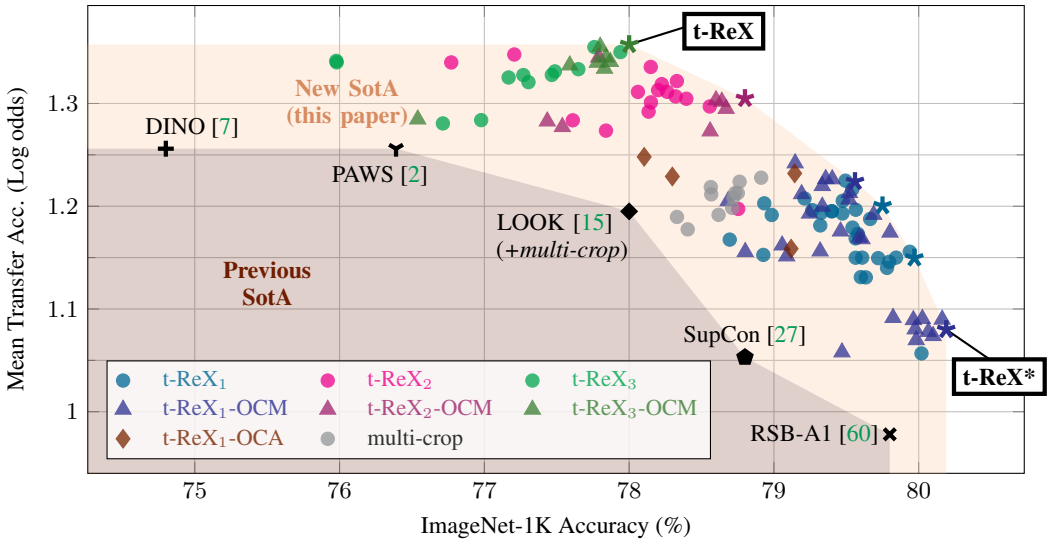


Figure 5: **Comparison on the training task vs transfer task performance for ResNet50.** We report IN1K (Top-1 accuracy) and transfer performance (log odds) averaged over 13 datasets (5 concept generalization datasets [50], Aircraft [36], Cars196 [32], DTD [11], EuroSAT [22], Flowers [40], Pets [42], Food101 [4] and SUN397 [62]) for a large number of our models trained with the supervised training setup presented in Section 3. Models on the convex hull are denoted by stars. We compare to public state-of-the-art (SotA) models: the supervised RSB-A1 [60] and SupCon [27] models, the self-supervised DINO [7], the semi-supervised PAWS [2], and a variant of LOOK [15] using multi-crop.

performance of multi-crop; however, the average transfer performance is decreased compared to multi-crop without a projector. More hidden layers seem to increase transfer performance, at the cost of a decrease in IN1K accuracy. The same can be said about the dimension of the hidden layer in Table 1b. We see however that a larger d_h can significantly increase transfer performance, and moderately decrease IN1K accuracy. Finally, Table 1c and Table 1d show that the bottleneck dimension d_b and input ℓ_2-normalization have only a small influence on IN1K and transfer performance.

We then study how the number of local crops affects performance for different number of hidden layers L. Similar to our observations without projectors from the previous section, Figure 4b shows that using local crops significantly improves both IN1K and transfer performance.

4.2 Pushing the envelope of training-versus-transfer performance

In this section, we report and analyse results from more than 100 different models trained on IN1K, all different instantiations of the proposed training setup. We varied hyper-parameters such as the number of hidden layers in the expandable projector head or the training objective. The most important results are depicted in Figure 5, while an extended version of this figure is presented in Figure 10 in Appendix.

Previous state of the art. RSB-A1 [60] is a highly optimized supervised ResNet50 model with top performance on IN1K. The self-supervised DINO [7] model has shown top transfer learning performance, while also performing well on IN1K. The semi-supervised PAWS [2] model matches DINO in transfer performance, with improved IN1K accuracy. To our knowledge, RSB-A1 and DINO/PAWS are the current state-of-the-art ResNet50 models for IN1K classification and transfer learning respectively. We also compare to two recent models that do well on both IN1K and transfer: SupCon [27] and LOOK [15]. For all models except LOOK, we evaluate models provided by the authors. Since there was no code for LOOK, we reproduced the method and found it to perform significantly better when combined with multi-crop. We compare to this enhanced version.

Notation. Models trained with a projector with L hidden layers are reported as **t-ReX**$_L$. Given the importance of L, we use different colors for different values of L in Figure 5. We append **-OCA** and **-OCM** to the models that use the corresponding variant (see Section 3.3). In t-ReX-OCA and our implementation of LOOK, we use a projector with 1 hidden layers ($L = 1$), a hyper-parameter we

Table 2: **Original IN1K and IN1K-v2 results.** For each model, we report results on the original IN1K "Val" test set (the x-axis of Figure 5), as well as on the three test sets of the IN1K-v2 dataset [46], using in all cases the encoder, and the linear classifier trained on the original IN1K training set.

Model	original IN1K	IN1K-v2			
	Val	matched-frequency	threshold-0.7	top-images	Mean
DINO [7]	74.8	61.9	71.2	76.7	69.9
PAWS [2]	76.4	63.6	73.0	78.3	71.6
LOOK [15] + *multi-crop*	78.0	65.8	75.3	80.7	73.9
SupCon [27]	78.8	66.8	75.5	80.5	74.3
RSB-A1 [60]	79.8	68.1	76.6	81.6	75.4
t-ReX	78.0	65.6	74.9	80.2	73.6
t-ReX*	**80.2**	**69.0**	**77.5**	**82.0**	**76.2**

found to work best for these variants. Models using multi-crop but no expandable projector head are listed as **multi-crop**. Models on the "envelope" (i.e., the convex hull) of Figure 5 are highlighted with a star and the corresponding configuration parameters are detailed in Table 4 and Table 8 of Appendix.

Main results. IN1K and transfer results of a selected set of models obtained with our setup are shown in Figure 5. Our main observations can be summarized as follows.

- ***Pushing the envelope.*** Many variants from our supervised training setup "push" the envelope beyond the previous state of the art, across both axes. Several of these models improve over the state of the art on one or the other axis, but no single model outperforms all the others on both dimensions. As the number of hidden layers of the projector increases, models gradually move from the lower right to upper left corners of the plane. This shows that increasing the projector complexity increases transfer performance at the cost of IN1K (training task) performance.
- ***No reason for no supervision.*** A large number of supervised variants outperform the DINO method with respect to transfer learning, while also being significantly better on IN1K. We therefore show that training with label supervision does not necessarily require to sacrifice transfer learning performance and one should use label information if available.
- ***State-of-the-art IN1K performance with three components.*** A number of t-ReX$_1$ models outperform the highly optimized RSB-A1 on IN1K, essentially by using only three components over the "vanilla" supervised learning process that is considered standard practice: a) cosine softmax with temperature, b) multi-crop data augmentation, and c) an expendable projector.
- ***Training with class prototypes brings further gain.*** Given the same projector configuration, training models with the OCM training objective (Equation (4)) has a slight advantage over training them with the cosine cross-entropy loss (Equation (2)). We see that 4 of the 6 points on the convex hull in Figure 5 are t-ReX-OCM models. This suggests that using class prototypes is a viable alternative to learning class weights end-to-end. On the other hand, and although generally outperforming other NCA variants like SupCon or LOOK, we find that our t-ReX-OCA variants are outperformed by the t-ReX-OCM variants along both axes. A more detailed discussion of all these variants is provided in Appendix C.3.2.
- ***Introducing t-ReX and t-ReX*.*** We single out the two instantiations that respectively excel on the transfer learning and IN1K axes, i.e., **t-ReX$_3$-OCM** and **t-ReX$_1$-OCM**. We simply rename them **t-ReX** and **t-ReX***, respectively. We envision these two **t**ransferable **Re**sNet50 models and their corresponding training setups to serve as strong supervised baselines for future research on transfer learning and IN1K. See Table 4 in Appendix for an exhaustive list of **t-ReX** and **t-ReX*** specifications and hyper-parameters.

4.3 Results on IN1K-v2

Finally, we compare **t-ReX** and **t-ReX*** to the previous state of the art on IN1K-v2. As before, for each model, we use the trained encoder as a feature extractor, and we reuse the linear classifier trained on IN1K and apply it directly to the test images of the different sets of IN1K-v2. Table 2 presents our results. Looking at the mean top-1 accuracy over the three test sets of IN1K-v2 , we observe that **t-ReX*** also matches the performance of RSB-A1 on IN1K-v2, showing strong generalization capabilities. It also outperforms SupCon, DINO, and LOOK with multi-crop.

5 Conclusion

We present a supervised training setup that leverages components from self-supervised learning, and improves generalization without conceding on the performance of the original task, i.e., IN1K classification. We also show that substituting class weights with an online class mean over a small memory bank boosts performance even further. We perform extensive evaluations on 13 transfer tasks to analyze the generalization properties of our models and show that many variants push the envelope on the IN1K-transfer performance plane. This validates our intuition that image-level supervision, if available, can be beneficial to both IN1K classification and transfer tasks.

With three easy-to-code modifications over the vanilla supervised learning training script (cosine cross entropy, multi-crop augmentations and an expendable projector head), one can train a ResNet50 encoder that, in 100 epochs, achieves over 80% top-1 accuracy on IN1K and leads to higher transfer learning performance than the publicly available 800-epoch SupCon [27] model. Using a different configuration with a larger projector head, our model can instead outperform the top self- and semi-supervised methods on transfer, and at the same time achieve highly improved performance on IN1K.

Acknowledgements. This work was supported in part by MIAI@Grenoble Alpes (ANR-19-P3IA-0003), and the ANR grant AVENUE (ANR-18-CE23-0011).

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proc. ICKDDM*, 2019.

[2] M. Assran, M. Caron, I. Misra, P. Bojanowski, A. Joulin, N. Ballas, and M. Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *Proc. ICCV*, 2021.

[3] L. Beyer, X. Zhai, and A. Kolesnikov. Better plain vit baselines for imagenet-1k. *arXiv:2205.01580*, 2022.

[4] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – Mining discriminative components with random forests. In *Proc. ECCV*, 2014.

[5] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proc. ECCV*, 2018.

[6] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proc. NeurIPS*, 2020.

[7] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proc. ICCV*, 2021.

[8] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proc. ICML*, 2020.

[9] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020.

[10] X. Chen and K. He. Exploring simple siamese representation learning. In *Proc. CVPR*, 2021.

[11] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proc. CVPR*, 2014.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

[13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proc. ICML*, 2014.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. ICLR*, 2021.

[15] Y. Feng, J. Jiang, M. Tang, R. Jin, and Y. Gao. Rethinking supervised pre-training for better downstream transferring. In *Proc. ICLR*, 2022.

[16] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018.

[17] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Proc. NeurIPS*, 2004.

[18] P. Goyal, D. Mahajan, A. Gupta, and I. Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proc. ICCV*, 2019.

[19] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Proc. NeurIPS*, 2020.

[20] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. CVPR*, 2020.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[22] P. Helber, B. Bischke, A. Dengel, and D. Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *JSTAEORS*, 2019.

[23] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv:1606.08415*, 2016.

[24] E. Hoffer, I. Hubara, and D. Soudry. Fix your classifier: the marginal value of training the last weight layer. In *Proc. ICLR*, 2018.

[25] M. Huh, P. Agrawal, and A. Efros. What makes imagenet good for transfer learning? *arXiv:1608.08614*, 2016.

[26] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[27] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. In *Proc. NeurIPS*, 2020.

[28] B. Ko and G. Gu. Large-scale bilingual language-image contrastive learning. *arXiv:2203.14463*, 2022.

[29] A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting Self-Supervised Visual Representation Learning. In *Proc. CVPR*, 2019.

[30] S. Kornblith, T. Chen, H. Lee, and M. Norouzi. Why do better loss functions lead to less transferable features? In *Proc. NeurIPS*, 2021.

[31] S. Kornblith, J. Shlens, and Q. Le. Do better imagenet models transfer better? In *Proc. CVPR*, 2019.

[32] J. Krause, J. Deng, M. Stark, and F.-F. Li. Collecting a large-scale dataset of fine-grained cars. In *Proc. ICCV-W*, 2013.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NeurIPS*, 2012.

[34] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *MP*, 45(1), 1989.

[35] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *Proc. ICLR*, 2017.

[36] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv:1306.5151*, 2013.

[37] S. Marcel and Y. Rodriguez. Torchvision the machine-vision package of torch. In *Proc. ACM-ICM*, 2010.

[38] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *Proc. ECCV*, 2012.

[39] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *Proc. ICCV*, 2017.

[40] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proc. ICCVGIP*, 2008.

[41] M. Pándy, A. Agostinelli, J. Uijlings, V. Ferrari, and T. Mensink. Transferability estimation using bhattacharyya class separability. In *Proc. CVPR*, 2022.

[42] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *Proc. CVPR*, 2012.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proc. NeurIPS*, 2019.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *JMLR*, 12, 2011.

[45] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proc. ICCV*, 2019.

[46] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *Proc. ICML*, 2019.

[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3), 2015.

[48] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry. Do adversarially robust ImageNet models transfer better? In *Proc. NeurIPS*, 2020.

[49] M. B. Sariyildiz, R. G. Cinbis, and E. Ayday. Key protected classification for collaborative learning. *PR*, 2020.

[50] M. B. Sariyildiz, Y. Kalantidis, D. Larlus, and K. Alahari. Concept generalization in visual representation learning. In *Proc. ICCV*, 2021.

[51] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *Proc. CVPR-W*, 2014.

[52] L. N. Smith. Cyclical learning rates for training neural networks. In *Proc. WACV*, 2017.

[53] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proc. NeurIPS*, 2016.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1), 2014.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015.

[56] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. CVPR*, 2016.

[57] N. Tripuraneni, M. Jordan, and C. Jin. On the theory of transfer learning: The importance of task diversity. In *Proc. NeurIPS*, 2020.

[58] X. Wang, H. Fan, Y. Tian, D. Kihara, and X. Chen. On the importance of asymmetry for siamese representation learning. In *Proc. CVPR*, 2022.

[59] Y. Wang, S. Tang, F. Zhu, L. Bai, R. Zhao, D. Qi, and W. Ouyang. Revisiting the transferability of supervised pretraining: an mlp perspective. In *Proc. CVPR*, 2022.

[60] R. Wightman, H. Touvron, and H. Jégou. Resnet strikes back: An improved training procedure in timm. In *Proc. NeurIPS-W*, 2021.

[61] Z. Wu, A. A. Efros, and S. X. Yu. Improving generalization via scalable neighborhood component analysis. In *Proc. ECCV*, 2018.

[62] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, 2010.

[63] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Proc. NeurIPS*, 2014.

[64] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proc. ICCV*, 2019.

[65] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proc. ICML*, 2021.

[66] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv:1910.04867*, 2019.

[67] H. Zhang, M. Cisse, Y. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. ICLR*, 2018.

[68] R. Zhang, P. Isola, and A. Efros. Colorful image colorization. In *Proc. ECCV*, 2016.

[69] N. Zhao, Z. Wu, R. W. H. Lau, and S. Lin. What makes instance discrimination good for transfer learning? In *Proc. ICLR*, 2021.

[70] W. Zheng, B. Zhang, J. Lu, and J. Zhou. Deep relational metric learning. In *Proc. ICCV*, 2021.

Appendix

Contents

A Further details on the improved supervised training setup

A.1 Data augmentations

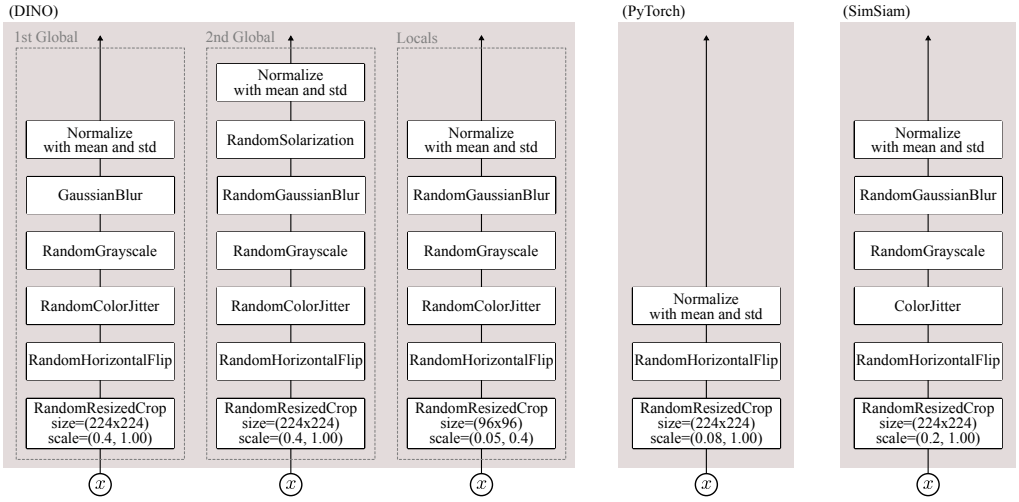
Our training setup for improving the generalization performance of supervised models (described in Section 3 of the main paper) includes the multi-crop augmentation initially proposed in SwAV [6]. In our experiments, we use the multi-crop implementation from DINO [7][2], which consists of three augmentation branches (two for global crops and one for local crops). The pipeline of this multi-crop augmentation is illustrated in Figure 6a. We also illustrate the pipelines of the "vanilla" PyTorch and SimSiam augmentations that are used in some of our ablation studies in Figure 6b and Figure 6c respectively. For our experiments with two global crops, i.e., when $M_g = 2$ in Figure 4a of the main paper, we use all the three branches. In all the other experiments, which have only one global crop $M_g = 1$, we use two of the branches: the second global crop and the one for local crops. Table 3 summarizes the parameters of the augmentation operations used in our experiments.

A.2 Details for OCA and OCM

In Section 3.3 of the main paper, we introduce "online" versions of the Nearest Class Mean classifier [38], referred to as Online Class Mean (OCM), and the Neighborhood Component Analysis [17], referred to as Online Component Analysis (OCA). In this section, we visualize and describe implementation details for the these models.

Illustrations of the different loss functions. In Figure 7 we visualize the supervised models which we train using the three loss functions defined in Section 3 of the main paper. Figure 7a, Figure 7b and Figure 7c correspond to the models for Equation (2), Equation (3) and Equation (4) of the main paper, respectively. As seen from Figure 7b and Figure 7c, and as explained in Section 3.3 of the

[2]The source code of DINO is released under Apache License 2.0.



(a) DINO augmentations [7] with multi-crop (b) PyTorch [37, 43] (c) SimSiam [10]

Figure 6: **Data-augmentation pipelines** considered in our work. We use the multi-crop augmentation implemented in DINO (a) as part of our improved training setup that is presented in Section 3 of the main paper. We also compare to the PyTorch (b) and SimSiam (c) augmentations in our ablations, see Appendix C.2. Note that the "no mc" baseline in Table 1 of the main paper corresponds to a supervised model trained with SimSiam augmentations instead of DINO's multi-crop. The parameters of the operations in (a) are given in Table 3. We use the default values for the operations in (b) and (c). These pipelines are implemented using torchvision [37] and Python Imaging Library. Accordingly, the operation and parameter names follow the conventions from these open-source libraries.

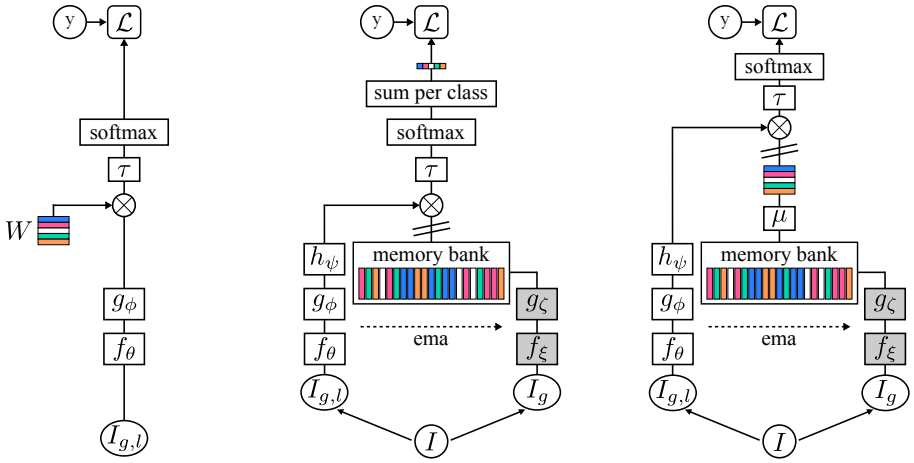
Table 3: **Default parameters of the DINO augmentations** used in our experiments. We implement the RandomGaussianBlur and RandomSolarization using Python Imaging Library, and use the torchvision [37] implementations for the remaining ones. For RandomGaussianBlur, "radius" denotes a range from which we uniformly sample radius values. Note that some of these operations involve other parameters, and in these cases we use their default values. The scale parameter of RandomResizedCrop is different for **t-ReX**, see Table 4 for details.

Augmentation Operation	Parameters
RandomResizedCrop for global crops	size=(224×224), scale=(0.4, 1.0)
RandomResizedCrop for local crops	size=(96×96), scale=(0.05, 0.4)
RandomHorizontalFlip	probability=0.5
RandomColorJitter	probability=0.8, brightness=0.4, contrast=0.4, saturation=0.2, hue=0.1
RandomGrayScale	probability=0.2
RandomGaussianBlur	probability=0.2, radius=(0.1, 2.0)
RandomSolarization	probability=0.2, threshold=128
Normalization	mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)

main paper, OCA and OCM follow SupCon [27] and LOOK [15] and use the momentum network and memory queue proposed in [20]. We explain them in detail below.

Momentum encoder f_ξ and projector g_ζ. To keep a memory bank of slowly evolving features, we keep an exponential moving average (EMA) of the encoder f_θ and projector g_ϕ parameters. Concretely, momentum encoder f_ξ and momentum projector g_ζ, whose parameters are respectively EMA of θ and ϕ, are defined as: $\xi \leftarrow m \times \xi + (1-m) \times \theta$ and $\zeta \leftarrow m \times \zeta + (1-m) \times \phi$, where $m = 0.999$ is the momentum parameter. As shown in Figures 7b and 7c, we only feed global crops I_g through these momentum networks f_ξ and g_ζ during training. Both global and local crops $I_{g,l}$ are passed through the encoder f_θ and projector g_ϕ.

Expendable predictor h_ψ after projector g_ϕ. In several SSL methods with dual-network architectures, e.g., BYOL [19] and SimSiam [10], breaking the architectural symmetry, by adding a multi-layer perceptron to one of the branches, was shown to improve the generalization of repre-



(a) For Equation (2) (b) For Equation (3), OCA (c) For Equation (4), OCM

Figure 7: **The supervised models** we train using our proposed setup. I_g and $I_{g,l}$ represent only global crops or both global and local crops. We refer to the models in (b) and (c) as OCA and OCM, respectively. Our **t-ReX** and **t-ReX*** variants have the form shown in (c), with projector configurations $L = 1$ (for **t-ReX***) and $L = 3$ (for **t-ReX**), $d_h = 2048$, $d_b = 256$, input ℓ_2-normalization enabled, memory bank size $\mathcal{Q} = 8192$ and with no predictor, i.e., h_ψ is an identity mapping.

sentations. Following this practice, we also experimented with training OCM and OCA models optionally using an expendable predictor head $h_\psi : \mathbb{R}^{d_b} \to \mathbb{R}^{d_b}$ with parameters ψ, added after the projector head g_ϕ (as shown in Figures 7b and 7c). These predictor heads contain fully-connected, batch-normalization [26] and GeLU [23] layers, followed by another fully-connected layer and ℓ_2-normalization. The first (resp. second) fully-connected layer maps from $\mathbb{R}^{d_b}$ to $\mathbb{R}^{d_p}$ (resp. $\mathbb{R}^{d_p}$ to $\mathbb{R}^{d_b}$). In our experiments d_p is generally 2048 dimensions. See Appendix C.3.2 for a discussion on the impact of predictors for performance; Note that neither **t-ReX** nor **t-ReX*** is trained with an expendable predictor head h_ψ.

Memory bank. The original NCM [38] and NCA [17] formulations require access to the entire dataset at each SGD training iteration, which is not possible in our case. To circumvent this, we use a memory queue $\mathcal{Q}$ which stores ℓ_2-normalized momentum projector outputs $\mathcal{Q} = \{g_\zeta(f_\xi(I_g))\}$ for global crops. In OCM, we compute a "prototype" for each class c, as the mean over all memory points from that class $\mu_c = {}^1/_{N_c} \sum_{z \in \mathcal{Q}_c} z$, where $\mathcal{Q}_c$ denotes samples in the queue that belong to class c and $N_c = |\mathcal{Q}_c|$. Then, for a given training crop I_j (can be either a global or local crop), we compute its predictor outputs $h_\psi(g_\phi(f_\theta(I_j)))$ and obtain class prediction scores by taking the inner product between this predictor output and the set of all class prototypes as defined in Equation (4) of the main paper. In OCA, we compute pairwise scores between the predictor output of the training crop $h_\psi(g_\phi(f_\theta(I_j)))$ and all the memory points $\mathcal{Q}$ to compute negative log NCA probabilities as in Equation (3) of the main paper.

B Further details on the evaluation process

B.1 Hyper-parameters for IN1K training

As we discuss in the previous section, we train supervised models on IN1K with different objectives and architecture configurations. We observe that the hyper-parameters for IN1K training that we share in Table 4 work well for the most effective models we studied.

B.2 Evaluation datasets

Once we train our models on IN1K, we evaluate their encoder representations $f_\theta(I)$ by training linear logistic regression (Log.Reg.) classifiers on 13 transfer datasets which include 5 ImageNet-CoG levels [50] and 8 commonly used small-scale datasets: Aircraft [36], Cars196 [32], DTD [11],

Table 4: **Hyper-parameters for training** our models on IN1K. Hyper-parameters shared by all models are given on the top part while the ones specific to **t-ReX** and **t-ReX*** are shown on the bottom part. Note that neither **t-ReX** nor **t-ReX*** is trained with an expendable predictor head h_ψ.

Configuration	Value for all models	
Optimizer	SGD	
Base learning rate	0.1	
Learning rate rule	$0.1 \times {}^{\text{batch size}}/_{256}$	
Learning rate warmup	Linear, 10 epochs	
Learning rate decay rule	Cosine schedule [35]	
Weight decay	0.0001	
Momentum	0.9	
# GPUs	4	
Batch size per GPU	64	
Batch size total	256	
Epochs	100	
Synchronize batch norms across GPUs	✓	
Mixed precision	✓	
τ in Equations (2, 3 and 4) of the main paper	0.1	
Augmentation pipeline from	DINO [7]	
# Global crops (M_g)	1	
# Local crops (M_l)	8	
Global crop resolution	224	
Global crop scale range	(0.4, 1)	
Local crop resolution	96	
Local crop scale range	(0.05, 0.4)	
	Value for **t-ReX**	Value for **t-ReX***
Projector input ℓ_2-norm	✓	✓
Projector L	3	1
Projector d_h	2048	2048
Projector d_b	256	256
Global crop scale range	(0.25, 1)	(0.4, 1)
Local crop scale range	(0.05, 0.25)	(0.05, 0.4)
Memory bank size $\lvert\mathcal{Q}\rvert$	8192	8192
Loss function used for training	$\mathcal{L}^*_{\text{OCM}}$	$\mathcal{L}^*_{\text{OCM}}$

EuroSAT [22], Flowers [40], Pets [42], Food101 [4] and SUN397 [62]. Additionally, we test the generalization of models to IN1K concepts using the three test sets of IN1K-v2 [46]. Statistics of all these datasets are provided in Table 5.

When a `val` split is not provided for a dataset, we randomly split its `train` set into two, following the size of `train` and `val` splits from either [15] or [19]. We also created different `train`/`val` splits when tuning hyper-parameters with different seeds, thus further increasing the robustness of our scores. Other notes on the datasets are as follows:

- **(i)** For DTD [11] (resp. EuroSAT [22]), there are 10 official `train`/`val`/`test` (resp. `train`/`test`) splits. Following [15, 19], we use the first split.
- **(ii)** For EuroSAT [22], we are not aware of either an official dataset split or the exact splits used in prior work, e.g., in [15]. So, we create random `train`/`val`/`test` splits following the number of samples in each split from [15], ensuring that the `val` and `test` splits are balanced to contain the same number of samples for each class.
- **(iii)** We use the Log.Reg. classifier trained on IN1K for predicting image labels in the three test sets of IN1K-v2. This is because IN1K-v2 is only composed of three test sets and no training data is provided.

Table 5: **Datasets** used for training (IN1K) and evaluating (the others) the quality of visual representations. We report top-1 accuracy for each dataset. Further implementation details on the utilization of the datasets are in Appendix A.1. *CCAS-4.0* denotes the Creative Commons Attribution-ShareAlike 4.0 international license.

Dataset	# Classes	# Train samples	# Val samples	# Test samples	Val provided	Test provided	License
For training models							
IN1K	1000	1281167	–	50000	–	✓	Research-only
For evaluating models on IN1K concepts							
IN1K-v2 [46]	1000	–	–	3×10000	–	✓	Research-only
For evaluating models on transfer tasks							
CoG L_1 [50]	1000	895359	223445	50000	–	✓	Research-only
CoG L_2 [50]	1000	892974	222814	50000	–	✓	Research-only
CoG L_3 [50]	1000	876495	218708	50000	–	✓	Research-only
CoG L_4 [50]	1000	886013	221115	50000	–	✓	Research-only
CoG L_5 [50]	1000	873630	218024	50000	–	✓	Research-only
Aircraft [36]	100	3334	3333	3333	✓	✓	Research-only
Cars196 [32]	196	5700	2444	8041	–	✓	Research-only
DTD [11]	47	1880	1880	1880	✓	✓	*Unclear*
EuroSAT [22]	10	13500	5400	8100	–	–	Research-only
Flowers [40]	102	1020	1020	6149	✓	✓	*Unclear*
Pets [42]	37	2570	1110	3669	–	✓	CCAS-4.0
Food101 [4]	101	68175	7575	25250	–	✓	*Unclear*
SUN397 [62]	397	15880	3970	19850	–	✓	Research-only

B.3 Evaluation metrics: the average log odds transferability score

Following Kornblith *et al.* [31], we compute log odds over all transfer datasets and use their average as a *transferability score*. This is the main metric we report in the different plots of the main paper. Denoting n_{correct} and $n_{\text{incorrect}}$ as the number of correct and incorrect predictions for a dataset, we compute the accuracy p and log odds score as follows:

$$p = \frac{n_{\text{correct}}}{n_{\text{correct}} + n_{\text{incorrect}}}, \quad \text{log odds} = \log \frac{p}{1-p}. \tag{5}$$

Then we report log odds averaged over all transfer datasets. See Appendix C.1 for per-dataset top-1 accuracies and average log odd scores for the models we compare in the main paper.

B.4 Evaluation protocols

We perform image classification on each evaluation dataset with logistic regression (Log.Reg.) classifiers following one of the two protocols proposed in [50] (for the 5 CoG levels) or in [31] (for the 8 small-scale datasets). In all cases, we first extract and store a (single) feature vector for each image and then learn the Log.Reg. classifiers on top of those features. Our classifiers are therefore trained *without data augmentation*, and this is why we report lower performance for the RSB model than the one presented in [60]. We extract image representations from the encoders f_θ by resizing an image with bicubic interpolation such that its shortest side is 224 pixels and then taking a central crop of size 224×224 pixels. The protocols for training Log.Reg. classifiers on the 5 CoG levels and on the other 8 datasets are different and detailed below.

Log.Reg. on the ImageNet-CoG levels. We apply ℓ_2-normalization to the pre-extracted features using the publicly-available source code of [50], and then train Log.Reg. classifiers using SGD with momentum = 0.9 and batch size = 1024 for 100 epochs. To treat each model as fairly as possible, we set the learning rate and weight decay hyper-parameters using `train`/`val` splits (`val` splits are randomly sampled using 20% of the original `train` splits). We use Optuna [1] and sample 30 different pairs. We train the final classifiers with these hyper-parameters on the union of the `train` and `val` splits, and report top-1 accuracy on the `test` splits. We repeat this process 5 times with different seeds and report averaged results.

Table 6: **Hyper-parameters for evaluating** our **t-ReX** and **t-ReX*** models using Log.Reg. classifiers on transfer datasets found by Optuna [1]. "Learning rate" and "Weight decay" are the parameters of the SGD optimizer used for training Log.Reg. classifiers as in the ImageNet-CoG protocol [50]. C is the inverse regularization coefficient used when training Log.Reg. classifiers with L-BFGS implemented in Scikit-learn [44].

	Configuration	*IN1K*	*CoG* L_1	*CoG* L_2	*CoG* L_3	*CoG* L_4	*CoG* L_5	*Aircraft*	*Cars196*	*DTD*	*EuroSAT*	*Flowers*	*Pets*	*Food101*	*SUN397*
t-ReX	Learning rate	12.8	11.0	10.8	11.0	11.0	11.0	–	–	–	–	–	–	–	–
	Weight decay	2.4e-10	1.1e-8	7.3e-10	1.1e-8	1.1e-8	1.1e-8	–	–	–	–	–	–	–	–
	C	–	–	–	–	–	–	42169	5109	1.1	9.1	14678	1778	0.4	1.1
t-ReX*	Learning rate	16.9	23.6	28.6	21.4	75.3	75.3	–	–	–	–	–	–	–	–
	Weight decay	1.2e-8	6.3e-10	4.3e-9	1.8e-9	4e-8	4e-8	–	–	–	–	–	–	–	–
	C	–	–	–	–	–	–	42169	14667	1.1	75	14678	42169	3.2	3.2

Table 7: **Compared models** with public ResNet50 encoder weights trained on IN1K by the authors.

Model	Epochs	Additional Notes	Repository URL	License
DINO [7]	800	Self-supervised	https://github.com/facebookresearch/dino	Apache-2.0
Barlow Twins [65]	1000	Self-supervised	https://github.com/facebookresearch/barlowtwins	MIT
PAWS [2]	300	Semi-supervised (With 10% of annotations)	https://github.com/facebookresearch/suncet	MIT
SupCon [27]	800	Supervised (with momentum encoder and memory bank)	https://github.com/HobbitLong/SupContrast	BSD-2-Clause
RSB-A1 [60]	600	Supervised	https://github.com/rwightman/pytorch-image-models	Apache-2.0

Log.Reg. on the smaller-scale transfer datasets. Following [31], we train Log.Reg. classifiers with pre-extracted features using L-BFGS [34]. To this end, we use the implementation in Scikit-learn [44]. We set the inverse regularization coefficient ("C") on each dataset using Optuna and their `train`/`val` splits over 25 trials. If a dataset does not have a fixed validation set, then we repeat hyper-parameter selection 5 times with different seeds.

B.5 Hyper-parameters for t-ReX and t-ReX* on transfer datasets

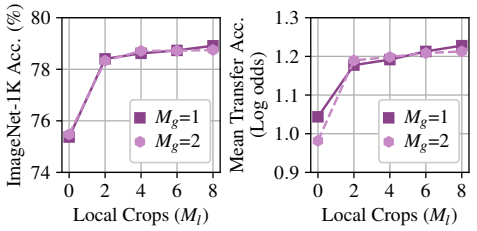
In Table 6 we present the hyper-parameters found by Optuna [1] for the Log.Reg. classifiers trained on the transfer datasets for **t-ReX** and **t-ReX***.

B.6 List of publicly available pretrained models used for comparisons

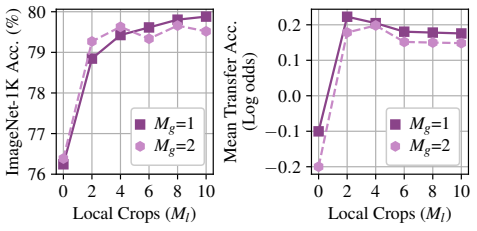
In Section 4.2 of the main paper, we compare our models to several prior works which are state-of-the-art either for IN1K classification or for transfer learning. These include self-supervised DINO [7], semi-supervised PAWS [2] and fully-supervised SupCon [27], RSB-A1 [60] and LOOK [15]. Additionally, we evaluated another self-supervised model Barlow Twins [65] but found it to be inferior to DINO on both IN1K and transfer datasets. We do not show these results in the main paper, but have included them here. For all the models except LOOK, we evaluate the best ResNet50 encoders trained on IN1K by their respective authors. Since there was neither publicly-available model nor source code for LOOK, we reproduced the method and found it to perform significantly better when combined with multi-crop. We compare to this enhanced version which we call LOOK+*multi-crop*. In Table 7, we give a list of the compared models with public encoder weights.

Table 8: **Top-1 Log.Reg. accuracy per dataset.** Mean LO is average log odds computed over all transfer datasets (i.e., all datasets except IN1K). In the main paper, we only plot IN1K and Mean LO scores for each model. For the datasets without a fixed validation set (see Appendix B.2) we repeat each evaluation 5 times with different seeds; variance is generally negligible. We bold the highest number for each column.

Model	*IN1K*	*CoG L_1*	*CoG L_2*	*CoG L_3*	*CoG L_4*	*CoG L_5*	*Aircraft*	*Cars196*	*DTD*	*EuroSAT*	*Flowers*	*Pets*	*Food101*	*SUN397*	*Mean LO*
Previous SotA															
DINO [7]	74.8	71.1	67.2	63.2	62.6	**57.6**	62.5	67.4	**77.7**	**97.7**	95.6	88.9	78.7	66.0	1.256
PAWS [2]	76.4	71.2	67.3	63.1	62.1	56.6	63.2	71.6	76.2	96.9	95.8	91.2	77.5	65.4	1.256
LOOK+*multi-crop* [15]	78.0	70.2	65.9	61.7	60.4	54.7	62.4	71.1	73.5	96.3	94.9	93.3	75.1	64.1	1.195
SupCon [27]	78.8	69.9	64.7	60.6	59.1	53.1	57.3	60.9	74.6	95.7	91.6	92.8	71.9	62.8	1.053
RSB-A1 [60]	79.8	69.9	65.0	60.9	59.3	52.8	47.1	54.0	73.9	95.7	88.7	93.1	71.2	63.3	0.978
Our models on the convex hull in Figure 5 of the main paper															
t-ReX	78.0	72.0	**68.3**	**63.9**	**63.4**	57.2	**67.3**	**74.2**	**77.7**	97.5	**96.2**	92.6	**80.1**	66.7	**1.357**
t-ReX-OCM (L=2, $\mathcal{Q}$|=8K)	78.8	**72.3**	68.2	63.7	63.0	56.8	64.7	70.8	75.8	97.3	95.3	93.2	79.1	**66.9**	1.305
t-ReX-OCM (L=1, h_ψ, |$\mathcal{Q}$|=131K)	79.6	71.7	67.3	62.8	61.6	55.3	61.9	68.8	75.2	96.7	94.0	**93.6**	76.6	66.1	1.224
t-ReX$_1$ (ℓ_2, L=1, d_h=4096, d_b=256)	79.8	71.7	67.1	63.0	61.8	54.8	61.1	66.7	74.4	96.8	93.2	93.5	76.7	66.2	1.201
t-ReX$_1$ (ℓ_2, L=1, d_h=2048, d_b=256)	80.0	71.3	66.4	62.3	60.6	53.9	58.8	67.5	75.2	96.4	91.6	93.4	75.4	65.4	1.150
t-ReX*	**80.2**	70.7	66.0	61.5	59.8	53.4	55.5	64.7	73.2	96.2	90.1	93.0	73.2	64.8	1.078



(a) Using **vanilla** softmax, i.e., encoder features x and class weights W are not ℓ_2-normalized and $\tau = 1.0$ in Equation (2) of the main paper.



(b) Using **cosine** softmax, following Equation (2) of the main paper, i.e., encoder features x and class weights W are ℓ_2-normalized and $\tau = 0.1$.

Figure 8: **Ablating the number of global and local crops for different softmax losses** without using a projector head, i.e., g_ϕ is an identity mapping in Equation (2) of the main paper. Figure (a) corresponds to Figure 4a of the main paper.

C Extended results and evaluations

C.1 Results per dataset

In Table 8, we report top-1 accuracy on each transfer dataset and on IN1K. Results are obtained by Log.Reg. classifiers, for all the models listed in Table 7 and for the ones that belong to the "convex hull" or envelope, denoted by stars in Figure 5 of the main paper.

C.2 Extended multi-crop ablations

In order to ablate multi-crop independently, i.e., without also using a projector head, in Figure 4a of the main paper we report results for the case where the projector head g_ϕ is an identity mapping. For these experiments, we train models using "vanilla" softmax, i.e., encoder features x and class weights W are not ℓ_2-normalized and $\tau = 1$ in Equation (2) of the main paper. The reason for this is that cosine softmax, i.e., Equation (1) shown in the main paper, suffers from overfitting to IN1K, and yields much worse transfer performance, as we show in Figure 8. Note that this phenomenon is also observed in [30] and explained by the fact that cosine softmax increases class separability of seen concepts in the feature space which reduces transferability. Therefore, we believe that cosine softmax is clearly sub-optimal for such a study. We note, however, that, as we show in Section 4 of the main paper, this overfitting of cosine softmax is alleviated by using projector heads.

Table 9: **Varying the scale of global and local crops.** We train models using different minimum and maximum scales for global and local crops. M_l is the number of local crops. We use 1 global crop, i.e., $M_g = 1$, for each experiment. Note that for the experiments presented in this table we train models using vanilla softmax, see Appendix C.2 for a discussion.

Augmentation Pipeline From	Global Scale	Local Scale	M_l	Epoch	IN1K	Mean Transfer
PyTorch [43]	(0.08, 1.00)	–	–	100	76.0	1.07
SimSiam [10]	(0.20, 1.00)	–	–	100	76.0	1.06
DINO [7]	(0.15, 1.00)	(0.05, 0.15)	8	100	78.4	1.19
DINO [7]	(0.25, 1.00)	(0.05, 0.25)	8	100	78.6	1.21
DINO [7]	(0.40, 1.00)	(0.05, 0.40)	8	100	78.9	1.23
DINO [7]	(0.05, 1.00)	–	–	800	76.5	1.13

In the main paper, we use (0.05, 0.4) and (0.4, 1.0) as the scale range for local and global crops, i.e., we sample scale values from these intervals. Table 9 provides an ablation on the maximum and minimum scales for local and global crops, respectively. We evaluate 3 different values (0.15, 0.25 and 0.4) for the maximum scale of local crops, which is also set as the minimum scale for global crops. Although the results are comparable, we see that 0.40 produces slightly better results.

We further verified if the improvements from local crops are due to the fact that models are trained with more images. To test this, we trained a model for 800 epochs using single crops with scale range (0.05, 1.0), and observed that it performs comparably to training models without local crops, i.e., it barely improves over training models with PyTorch and SimSiam augmentations.

C.3 Extended set of results

C.3.1 The t-ReX$_L$-*orth* variant

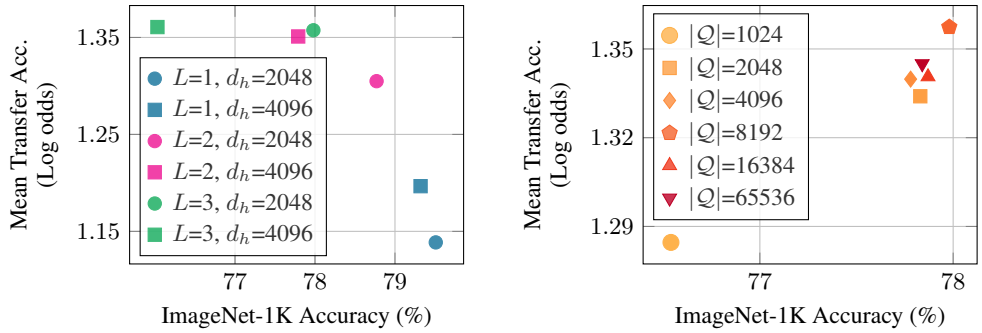
In Section 3.3 of the main paper, we propose the OCM variant where class weights W are replaced by class means $\{\mu_c\}_{c=1}^{C}$ which are obtained in an online manner using a memory bank (see also Appendix A.2 for details). Inspired by [24, 49], we explore another variant of our setup where class weights W are initialized with random *orthogonal* vectors and kept frozen while the encoder f_θ and projector g_ϕ networks are trained. Our motivation is that such vectors may lead to higher class separation obtaining higher accuracy on the training task, as also noted in [30].

To this end, without using momentum networks, a predictor head or a memory bank, we simply optimize Equation (2) in the main paper with "fixed" class weights, and denote these models with the t-ReX$_L$-*orth* notation. Due to our limited computational budget, we train 6 t-ReX$_L$-*orth* variants with and without projector input ℓ_2-norm, $L = [1, 2, 3]$, $d_h = 2048$ and $d_b = 256$, and plot their IN1K and average transfer accuracies along with the others from the main paper in Figure 10. We observe that none of those six models were able to expand the envelope of training-versus-transfer performance.

C.3.2 Discussion on t-ReX-OCA and t-ReX-OCM

t-ReX-OCA models achieve a balance between IN1K and transfer performances, i.e., they are towards the middle of the envelope in Figure 5 of the main paper and in Figure 10. Moreover, in our evaluations we see that these variants are better overall than LOOK+*multi-crop*. Note that the main difference between LOOK+*multi-crop* and the t-ReX-OCA variants is that the former restricts the soft k-NN loss [17] to the close neighborhood of each training sample in the memory bank. Our observation suggests that relying on more points in the memory bank is beneficial for learning better representations.

Varying the projector and predictor head architecture. In Table 1 of the main paper, we study the impact of the projector head parameters for the models we train with Equation (2). Here, we replicate this study for the OCM variant. Specifically, we ablate the L and d_h parameters for the OCM model defined by Equation (4). We show the results in Figure 9a, and see that our observation still holds, i.e., increasing the complexity of projector improves transfer performance at the cost of IN1K performance. We also tested the impact of predictors in OCM models, and observed a similar behavior: Small predictors improve the transferability of encoder representations by sacrificing IN1K performance; allowing



(a) Projector parameters L and d_h

(b) Memory size ($|\mathcal{Q}|$)

Figure 9: **OCM ablations** for (a) the projector head parameters L and d_h when $|\mathcal{Q}| = 8192$ and (b) the size of memory bank $|\mathcal{Q}|$ when $L = 3$ and $d_h = 2048$.

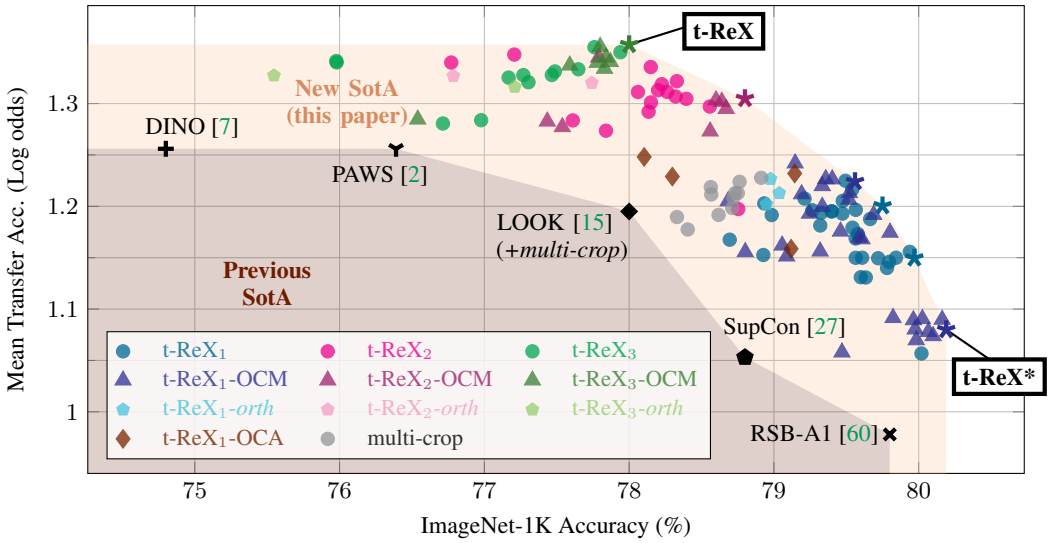


Figure 10: **Extended version of Figure 5 of the main paper, featuring the t-ReX$_L$-*orth* variant.** We plot six additional t-ReX$_L$-*orth* variants with and without projector input ℓ_2-norm, $L = [1, 2, 3]$, i.e., six variants of our **t-ReX** model. We report IN1K (top-1 accuracy) and transfer performance (log odds) averaged over 13 datasets (see Table 5 for the list) for a large number of our models trained with the supervised training setup presented in Section 3 of the main paper. Models on the convex hull are denoted by stars. We compare to public state-of-the-art (SotA) models: the supervised RSB-A1 [60] and SupCon [27] models, the self-supervised DINO [7], the semi-supervised PAWS [2], and a variant of LOOK [15] using multi-crop.

these variants to move along the envelope. Consequently, training OCM models with no predictor, i.e., when h_ψ is an identity mapping in Figure 7c, improves IN1K performance. Indeed, as explained in Section 3.3, our best model on IN1K, i.e., **t-ReX***, is a **t-ReX$_1$-OCM** variant with no predictor head.

Varying the memory bank size. In our experiments, we observe that the size of the memory bank $|\mathcal{Q}|$ does not severely affect the performance of the OCA or OCM variants. Figure 9b shows how the size of the memory bank impacts performance for the OCM variant, where we see that even a moderately-sized memory bank of $|\mathcal{Q}| = 8192$, i.e., containing only 8 points per class on average is sufficient to obtain high performance. However, we note that for the OCA variants, it is important to tune the temperature parameter τ for a given memory bank size to control the smoothness of NCA probabilities computed in Equation (3) of the main paper.

D Limitations

Requirement for annotations. Firstly, our training setup is tailored for supervised learning, and therefore, its performance on both training and transfer tasks depends on the availability of high-quality and diverse annotations for the training images. Image-level annotation, when involving a large number of potentially fine-grained classes is an expensive and error-prone process. In this work, we show that given a large-scale *curated* and annotated dataset, more precisely given IN1K, which is composed of 1.28M images annotated for 1000 different concepts, it is possible to learn more generic representations than self- or semi-supervised models. When only a handful of concepts is annotated, or when annotations contain too much noise, these conclusions might not be accurate anymore, and both the pretraining classification task and the transfer tasks results might be degraded. In this case, self- or semi-supervised approaches might become more relevant. Yet, those scenarios are out of the scope of our study.

Specific encoder architecture. Secondly, we develop our training setup based on a single encoder architecture, ResNet50, and do not test it on other architectures. This was motivated by the fact that ResNet50 is still a very commonly used architecture. Also, we note that our training setup components, multi-crop, a data-augmentation operator, and the expendable projector, added after the encoder, are architecture-agnostic so those contributions can be seamlessly applied to any other architecture of choice. Therefore, it is reasonable to expect that our setup would consistently improve other architecture families, such as Vision Transformers (ViTs) [14]. In fact, both components were previously successfully used with ViTs for self-supervised learning [7]. We leave studying the applicability of our training setup to other encoder architectures as future work.