arXiv:2104.14548v2 [cs.CV] 7 Oct 2021

With a Little Help from My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations

Debidatta Dwibedi [1], Yusuf Aytar [2], Jonathan Tompson [1], Pierre Sermanet [1], and Andrew Zisserman [2]

[1] Google Research, [2] DeepMind

{debidatta, yusufaytar, tompson, sermanet, zisserman}@google.com

Abstract

*Self-supervised learning algorithms based on instance discrimination train encoders to be invariant to pre-defined transformations of the **same** instance. While most methods treat different views of the same image as positives for a contrastive loss, we are interested in using positives from **other** instances in the dataset. Our method, Nearest-Neighbor Contrastive Learning of visual Representations (NNCLR), samples the nearest neighbors from the dataset in the latent space, and treats them as positives. This provides more semantic variations than pre-defined transformations.*

We find that using the nearest-neighbor as positive in contrastive losses improves performance significantly on ImageNet classification, from 71.7% to 75.6%, outperforming previous state-of-the-art methods. On semi-supervised learning benchmarks we improve performance significantly when only 1% ImageNet labels are available, from 53.8% to 56.5%. On transfer learning benchmarks our method outperforms state-of-the-art methods (including supervised learning with ImageNet) on 8 out of 12 downstream datasets. Furthermore, we demonstrate empirically that our method is less reliant on complex data augmentations. We see a relative reduction of only 2.1% ImageNet Top-1 accuracy when we train using only random crops.

1. Introduction

How does one make sense of a novel sensory experience? What might be going through someone's head when they are shown a picture of something new, say a dodo? Even without being told explicitly what a dodo is, they will likely form associations between the dodo and other similar semantic classes; for instance a dodo is more similar to a chicken or a duck than an elephant or a tiger. This act of contrasting and comparing new sensory inputs with what one has already experienced happens subconsciously and might play a key role [25] in how humans are able to acquire concepts quickly. In this work, we show how an ability to find similarities across items within previously seen examples improves the performance of *self-supervised* representation learning.

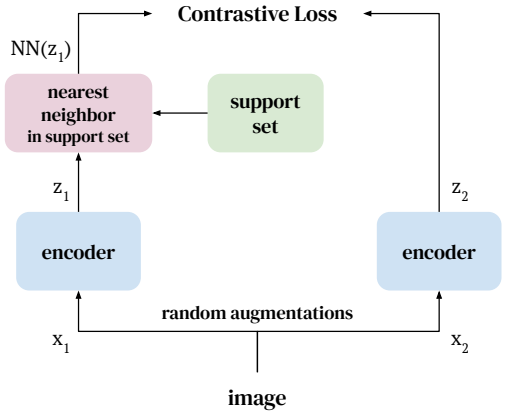


Figure 1: **NNCLR Training.** We propose a simple self-supervised learning method that uses similar examples from a support set as positives in a contrastive loss.

A particular kind of self-supervised training – known as instance discrimination [11, 33, 60] – has become popular recently. Models are encouraged to be invariant to multiple transformations of a *single* sample. This approach has been impressively successful [11, 28] at bridging the performance gap between self-supervised and supervised models. In the instance discrimination setup, when a model is shown a picture of a dodo, it learns representations by being trained to differentiate between what makes that specific dodo image *different* from everything else in the training set. In this work, we ask the question: if we empower the model to also find other image samples *similar* to the given dodo image, does it lead to better learning?

Current state-of-the-art instance discrimination methods generate positive samples using *data augmentation*, random

image transformations (e.g. random crops) applied to the same sample to obtain multiple views of the same image. These multiple views are assumed to be positives, and the representation is learned by encouraging the positives to be as close as possible in the embedding space, without collapsing to a trivial solution. However random augmentations, such as random crops or color changes, can not provide positive pairs for different viewpoints, deformations of the same object, or even for other similar instances within a semantic class. The onus of generalization lies heavily on the data augmentation pipeline, which cannot cover all the variances in a given class.

In this work, we are interested in going beyond *single instance positives*, i.e. the instance discrimination task. We expect by doing so we can learn better features that are invariant to different viewpoints, deformations, and even intra-class variations. The benefits of going beyond single instance positives have been established in [30, 39], though these works require class labels or multiple modalities (RGB frames and flow) to obtain the positives which are not applicable to our domain. Clustering-based methods [6, 8, 66] also offer an approach to go beyond single instance positives, but assuming the entire cluster (or its prototype) to be positives could hurt performance due to early over-generalization. Instead we propose using *nearest neighbors* in the learned representation space as positives.

We learn our representation by encouraging proximity between different views of the same sample and their nearest neighbors in the latent space. Through our approach, Nearest-Neighbour Contrastive Learning of visual Representations (NNCLR), the model is encouraged to generalize to new data-points that may not be covered by the data augmentation scheme at hand. In other words, nearest-neighbors of a sample in the embedding space act as small semantic perturbations that are not imaginary, i.e. they are representative of actual semantic samples in the dataset. We implement our method in a contrastive learning setting similar to [11, 12]. To obtain nearest-neighbors, we utilize a support set that keeps embeddings of a subset of the dataset in memory. This support set also gets constantly replenished during training. Note that our support set is different from memory banks [55, 60] and queues [13], where the stored features are used as negatives. We utilize the support set for nearest neighbor search for retrieving cross-sample positives. Figure 1 gives an overview of the method.

We make the following contributions: (i) We introduce NNCLR to learn self-supervised representations that go beyond single instance positives, without resorting to clustering; (ii) We demonstrate that NNCLR increases the performance of contrastive learning methods (e.g. SimCLR [12]) by $\sim$ 3.8% and achieves state of the art performance on ImageNet classification for linear evaluation and semi-supervised setup with limited labels; (iii) Our method outperforms state of the art methods on self-supervised, and even supervised features (learned via supervised ImageNet pre-training), on 8 out of 12 transfer learning tasks; Finally, (iv) We show that by using the NN as positive with only random crop augmentations, we achieve 73.3% ImageNet accuracy. This reduces the reliance of self-supervised methods on data augmentation strategies.

2. Related Work

Self-supervised Learning. Self-supervised representation learning aims to obtain robust representations of samples from raw data without expensive labels or annotations. Early methods in this field focused on defining *pre-text tasks* – which typically involves defining a surrogate task on a domain with ample weak supervision labels, like predicting the rotation of images [27], relative positions of patches in an image [17], or tracking patches in a video [49, 58]. Encoders trained to solve such pre-text tasks are expected to learn general features that might be useful for other downstream tasks requiring expensive annotations (e.g. image classification).

One broad category of self-supervised learning techniques are those that use contrastive losses, which have been used in a wide range of computer vision applications[10, 29, 51]. These methods learn a latent space that draws positive samples together (e.g. adjacent frames in a video sequence), while pushing apart negative samples (e.g. frames from another video). In some cases, this is also possible without explicit negatives [28]. More recently, a variant of contrastive learning called *instance discrimination* [11, 13, 20, 60] has seen considerable success and have achieved remarkable performance [8, 11, 12, 13] on a wide variety of downstream tasks. They have closed the gap with supervised learning to a large extent. Many techniques have proved to be useful in this pursuit: data augmentation, contrastive losses [11, 12, 33], momentum encoders [13, 28, 33] and memory banks [13, 55, 60]. In this work, we extend instance discrimination to include non-trivial positives, not just between augmented samples of the same image, but also from among different images. Methods that use prototypes/clusters [1, 3, 6, 7, 8, 26, 36, 59, 62, 63, 64, 66] also attempt to learn features by associating multiple samples with the same cluster. However, instead of clustering or learning prototypes, we maintain a support set of image embeddings and using nearest neighbors from that set to define positive samples.

Queues and Memory Banks. In our work, we use a support set as memory during training. It is implemented as a queue similar to MoCo [33]. MoCo uses elements of the queue as negatives, while this work uses nearest neighbors in the queue to find positives in the context of contrastive losses. [60] use a memory bank to keep a running average of embeddings of all the samples in the dataset. Likewise,

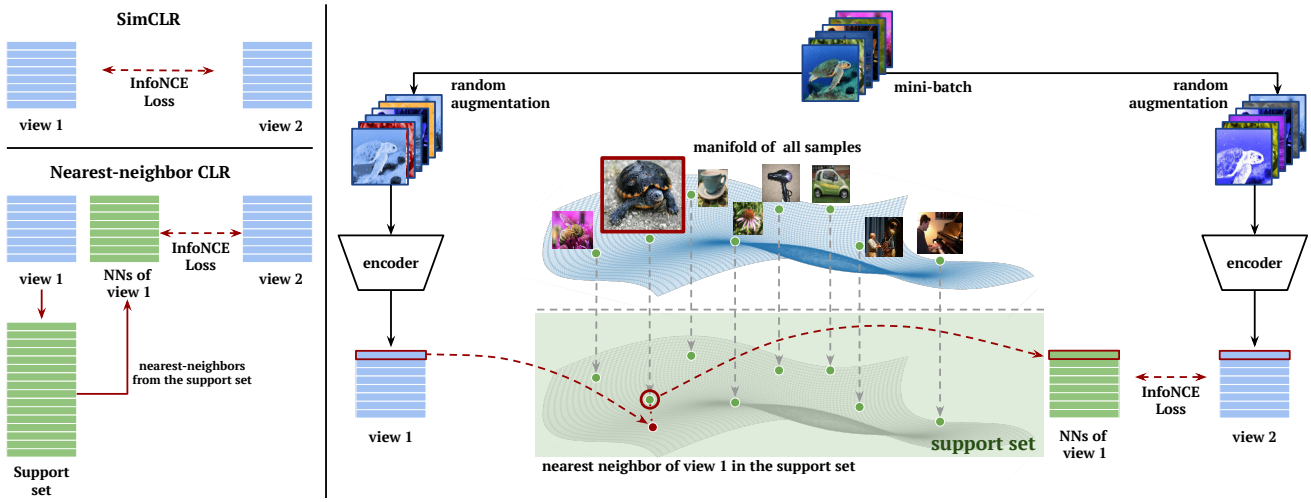


Figure 2: **Overview of NNCLR Training**

[66] maintains a clustered set of embeddings and uses nearest neighbors to those aggregate embeddings as positives. In our work, the size of the memory is fixed and independent of the training dataset, nor do we perform any aggregation or clustering in our latent embedding space. Instead of a memory bank, SwAV [8] stores prototype centers that it uses for clustering embeddings. SwAV's prototype centers are learned via training with Sinkhorn clustering and persist throughout pre-training. Unlike [8, 60, 66], our support set is continually refreshed with new embeddings and we do not maintain running averages of the embeddings.

Nearest Neighbors in Computer Vision. Nearest neighbor search has been an important tool across a wide range of computer vision applications [15, 18, 31, 32, 52, 59], from image retrieval to unsupervised feature learning. Nearest-neighbor lookup as an intermediate operation has also been useful for image alignment [54] and video alignment [21] tasks. [54] propose a method to learn landmarks on objects in an unsupervised manner by using nearest-neighbors from other images of the same object while [21] show unsupervised learning of action phases by using soft nearest-neighbors across videos of the same action. In our work, we also use cross-sample nearest neighbors but train on datasets with many classes of objects with the objective of learning transferable features. Related to our work, [30] uses nearest neighbor retrieval to define self-supervision for video representation learning across different modalities (e.g. RGB and optical flow). In contrast, in this work we use nearest neighbor retrieval within a single modality (RGB images), and we maintain an explicit support set of prior embeddings to increase diversity.

[2] concurrently propose leveraging nearest-neighbors in embedding space to improve self-supervised representation learning using BYOL [28]. The authors propose using an additional term for the nearest-neighbor to the BYOL loss. Instead, in our formulation all our loss terms use the nearest-neighbor.

3. Approach

We first describe constrastive learning (i.e. the InfoNCE loss) in the context of instance discrimination, and discuss SimCLR [11] as one of the leading methods in this domain. Next we introduce our approach, Nearest-Neighbor Contrastive Learning of visual Representations (NNCLR), which proposes using nearest-neighbours (NN) as positives to improve contrastive instance discrimination methods.

3.1. Contrastive instance discrimination

InfoNCE [47, 53, 60] loss (i.e. contrastive loss) is quite commonly used in the instance discrimination [11, 33, 60] setting. For any given embedded sample z_i, we also have another positive embedding z_i^+ (often a random augmentation of the sample), and many negative embeddings $z^- \in N_i$. Then the InfoNCE loss is defined as follows:

$$\mathcal{L}_i^{\text{InfoNCE}} = -\log \frac{\exp\left(z_i \cdot z_i^+/\tau\right)}{\exp\left(z_i \cdot z_i^+/\tau\right) + \sum_{z^- \in \mathcal{N}_i} \exp\left(z_i \cdot z^-/\tau\right)} \tag{1}$$

where (z_i, z_i^+) is the positive pair, (z_i, z^-) is any negative pair and τ is the softmax temperature. The underlying idea is learning a representation that pulls positive pairs together in the embedding space, while separating negative pairs.

SimCLR uses two views of the same image as the positive pair. These two views, which are produced using random data augmentations, are fed through an encoder to obtain the positive embedding pair z_i and z_i^+. The negative pairs (z_i, z^-) are formed using all the other embeddings in the given mini-batch.

Formally, given a mini-batch of images $\{x_1, x_2.., x_n\}$, two different random augmentations (or views) are gener-

ated for each image x_i, and fed through the encoder ϕ to obtain embeddings $z_i = \phi(\text{aug}(x_i))$ and $z_i^+ = \phi(\text{aug}(x_i))$, where $\text{aug}(\cdot)$ is the random augmentation function. The encoder ϕ is typically a ResNet-50 with a non-linear projection head. Then the InfoNCE loss used in SimCLR is defined as follows:

$$\mathcal{L}_i^{\text{SimCLR}} = -\log \frac{\exp\left(z_i \cdot z_i^+ / \tau\right)}{\sum_{k=1}^{n} \exp\left(z_i \cdot z_k^+ / \tau\right)} \tag{2}$$

Note that each embedding is l_2 normalized before the dot product is computed in the loss. Then the overall loss for the given mini-batch is $\mathcal{L}^{\text{SimCLR}} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i^{\text{SimCLR}}$.

As SimCLR solely relies on transformations introduced by pre-defined data augmentations on the same sample, it cannot link multiple samples potentially belonging to the same semantic class, which in turn might decrease its capacity to be invariant to large intra-class variations. Next we address this point by introducing our method.

3.2. Nearest-Neighbor CLR (NNCLR)

In order to increase the richness of our latent representation and go beyond single instance positives, we propose using nearest-neighbours to obtain more diverse positive pairs. This requires keeping a *support set* of embeddings which is representative of the full data distribution.

SimCLR uses two augmentations (z_i, z_i^+) to form the positive pair. Instead, we propose using z_i's nearest-neighbor in the support set Q to form the positive pair. In Figure 2 we visualize this process schematically. Similar to SimCLR we obtain the negative pairs from the mini-batch and utilize a variant of the InfoNCE loss (1) for contrastive learning. Building upon the SimCLR objective (2) we define NNCLR loss as below:

$$\mathcal{L}_i^{\text{NNCLR}} = -\log \frac{\exp\left(\text{NN}(z_i, Q) \cdot z_i^+ / \tau\right)}{\sum_{k=1}^{n} \exp\left(\text{NN}(z_i, Q) \cdot z_k^+ / \tau\right)} \tag{3}$$

where $\text{NN}(z, Q)$ is the nearest neighbor operator as defined below:

$$\text{NN}(z, Q) = \arg\min_{q \in Q} \|z - q\|_2 \tag{4}$$

As in SimCLR, each embedding is l_2 normalized before the dot product is computed in the loss (3). Similarly we apply l_2 normalization before nearest-neighbor operation in (4). We minimize the average loss over all elements in the mini-batch in order to obtain the final loss $\mathcal{L}^{\text{NNCLR}} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i^{\text{NNCLR}}$.

Implementation details. We make the loss symmetric [38, 50] by adding the following term to Eq. 3: $-\log\left(\exp\left(\text{NN}(z_i, Q) \cdot z_i^+ / \tau\right) / \sum_{k=1}^{n} \exp\left(\text{NN}(z_k, Q) \cdot z_i^+ / \tau\right)\right)$ Though, this does not affect performance emperically. Also, inspired from BYOL [28], we pass z_i^+ through a prediction head g to produce embeddings $p_i^+ = g(z_i^+)$. Then we use p_i^+ instead of z_i^+ in (3). Using a prediction MLP adds a small boost to our performance as shown in Section 4.4.

Support set. We implement our support set as a queue (i.e. first-in-first-out). The support set is initialized as a random matrix with dimension $[m, d]$, where m is the size of the queue and d is the size of the embeddings. The size of the support set is kept large enough so as to approximate the full dataset distribution in the embedding space. We update it at the end of each training step by taking the n (batch size) embeddings from the current training step and concatenating them at the end of the queue. We discard the oldest n elements from the queue. We only use embeddings from one view to update the support set. Using both views' embeddings to update does not lead to any significant difference in downstream performance. In Section 4.4 we compare the performance of multiple support set variants.

4. Experiments

In this section we compare NNCLR features with other state of the art self-supervised image representations. First, we provide details of our architecture and training process. Next, following commonly used evaluation protocol [11, 12, 28, 33], we compare our approach with other self-supervised features on linear evaluation and semi-supervised learning on the ImageNet ILSVRC-2012 dataset. Finally we present results on transferring self-supervised features to other downstream datasets and tasks.

4.1. Implementation details

Architecture. We use ResNet-50 [34] as our encoder to be consistent with the existing literature [11, 28]. We spatially average the output of ResNet-50 which makes the output of the encoder a 2048-d embedding. The architecture of the projection MLP is 3 fully connected layers of sizes $[2048, 2048, d]$ where d is the embedding size used to apply the loss. We use $d = 256$ in the experiments unless otherwise stated. All fully-connected layers are followed by batch-normalization [37]. All the batch-norm layers except the last layer are followed by ReLU activation. The architecture of the prediction MLP g is 2 fully-connected layers of size $[4096, d]$. The hidden layer of the prediction MLP is followed by batch-norm and ReLU. The last layer has no batch-norm or activation.

Training. Following other self-supervised methods [11, 12, 28, 33], we train our NNCLR representation on the ImageNet2012 dataset which contains $1,281,167$ images, without using any annotation or class labels. We train for 1000 epochs with a warm-up of 10 epochs with cosine anneal-

ing schedule using the LARS optimizer [65]. Weight-decay of 10^{-6} is applied during training. As is common practice [12, 28], we don't apply weight-decay to the bias terms. We use the data augmentation scheme used in BYOL [28] and we use a temperature τ of 0.1 when applying the softmax during computation of the contrastive loss in Equation 3. The best results of NNCLR are achieved with $98,304$ queue size and base learning rate [28] of 0.3.

4.2. ImageNet evaluations

ImageNet linear evaluation. Following the standard linear evaluation procedure [11, 28] we train a linear classifier for 90 epochs on the frozen 2048-d embeddings from the ResNet-50 encoder using LARS with cosine annealed learning rate of 1 with Nesterov momentum of 0.9 and batch size of 4096.

Comparison with state of the art methods is presented in Table 1. First, NNCLR achieves the best performance compared to all the other methods using a ResNet-50 encoder trained with two views. NNCLR provides more than 3.6% improvement over well known constrastive learning approaches such as MoCo v2 [13] and SimCLR v2 [12]. Even compared to InfoMin Aug. [56], which explicitly studies "good view" transformations to apply in contrastive learning, NNCLR achieves more than 2% improvement on top-1 classification performance. We outperform BYOL[28] (which is the state-of-the-art method among methods that use two views) by more than 1%.

We also achieve 3.6% improvement compared to the state of the art clustering based method SwAV [8] in the same setting of using two views. To compare with SwAV's multi-crop models, we pre-train for 800 epochs with 8 views (two 224×224 and six 96×96 views) using only the larger views to calculate the NNs. In this setting our method outperforms SwAV by 0.3% in Top-1 accuracy. Note that while multi-crop is responsible for 3.5% performance improvement for SwAV, for our method it provides a boost of only 0.2%. However, increasing the number of crops quadratically increases the memory and compute requirements, and is quite costly even when low-resolution crops are used as in [8].

Semi-supervised learning on ImageNet. We evaluate the effectiveness of our features in a semi-supervised setting on ImageNet 1% and 10% subsets following the standard evaluation protocol [12, 28]. We present these results in Table 2. The first key result of Table 2 is that our method outperforms all the state of the art methods on semi-supervised learning on ImageNet 1% subset, including SwAV's [8] multi-crop setting. This is a clear indication of good generalization capacity of NNCLR features, particularly in low-shot learning scenarios. Using the ImageNet 10% subset, NNCLR outperforms SimCLR [11] and other methods. However, SwAV's [8] multi-crop setting outperforms our method in ImageNet 10% subset.

Method	Top-1	Top-5
PIRL [45]	63.6	-
CPC v2 [35]	63.8	85.3
PCL [43]	65.9	-
CMC [55]	66.2	87.0
MoCo v2 [13]	71.1	-
SimSiam [14]	71.3	-
SimCLR v2 [12]	71.7	-
SwAV [8]	71.8	-
InfoMin Aug. [56]	73.0	91.1
BYOL [28]	74.3	91.6
NNCLR (ours)	**75.4**	**92.3**
SwAV (multi-crop) [8]	75.3	-
NNCLR (ours) (multi-crop)	**75.6**	**92.4**

Table 1: **ImageNet linear evaluation results.** Comparison with other self-supervised learning methods on ResNet-50 encoder. Methods on the top section use two views only.

Method	ImageNet **1%**		ImageNet **10%**	
	Top-1	Top-5	Top-1	Top-5
Supervised	25.4	48.4	56.4	80.4
InstDisc [60]	-	39.2	-	77.4
PIRL [45]	-	57.2	-	83.8
PCL [43]	-	75.6	-	86.2
SimCLR [11]	48.3	75.5	65.6	87.8
BYOL [28]	53.2	78.4	68.8	89.0
NNCLR (ours)	**56.4**	**80.7**	**69.8**	**89.3**
SwAV (multi-crop) [8]	53.9	78.5	70.2	89.9

Table 2: **Semi-supervised learning results on ImageNet.** Top-1 and top-5 performances are reported on fine-tuning a pre-trained ResNet-50 with ImageNet 1% and 10% datasets.

4.3. Transfer learning evaluations

We show representations learned using NNCLR are effective for transfer learning on multiple downstream classification tasks on a wide range of datasets. We follow the linear evaluation setup described in [28]. The datasets used in this benchmark are as follows: Food101 [5], CIFAR10 [42], CIFAR100 [42], Birdsnap [4], Sun397 [61], Cars [41], Aircraft [44], VOC2007 [22], DTD [16], Oxford-IIIT-Pets [48], Caltech-101 [23] and Oxford-Flowers [46]. Following the evaluation protocol outlined in [28], we first train a linear classifier using the training set labels while choosing the best regularization hyper-parameter on the respective validation set. Then we combine the train and validation set to create the final training set which is used to train the linear classifier that is evaluated on the test set.

Method	Food101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
BYOL [28]	75.3	91.3	78.4	57.2	62.2	**67.8**	60.6	82.5	75.5	90.4	94.2	**96.1**
SimCLR [28]	72.8	90.5	74.4	42.4	60.6	49.3	49.8	81.4	**75.7**	84.6	89.3	92.6
Sup.-IN [11]	72.3	93.6	78.3	53.7	61.9	66.7	61.0	82.8	74.9	91.5	**94.5**	94.7
NNCLR	**76.7**	**93.7**	**79.0**	**61.4**	**62.5**	67.1	**64.1**	**83.0**	75.5	**91.8**	91.3	95.1

Table 3: **Transfer learning performance** using ResNet-50 pretrained with ImageNet. For all datasets we report Top-1 classification accuracy except Aircraft, Caltech-101, Pets and Flowers for which we report mean per-class accuracy and VOC2007 for which we report 11-point MAP.

We present transfer learning results in Table 3. NNCLR outperforms supervised features (ResNet-50 trained with ImageNet labels) on 11 out of the 12 datasets. Moreover our method improves over BYOL [28] and SimCLR [11] on 8 out of the 12 datasets. These results further validate the generalization performance of NNCLR features.

4.4. Ablations

In this section we present a thorough analysis of NNCLR. After discussing the default settings, we start by demonstrating the effect of training with nearest-neighbors in a variety of settings. Then, we present several design choices such as support set size, varying k in top-k nearest neighbors, type of nearest neighbors, different training epochs, variations of batch size, and embedding size. We also briefly discuss memory and computational overhead of our method.

Default settings. Unless otherwise stated our support set size during ablation experiments is $32,768$ and our batch size is 4096. We train for 1000 epochs with a warm-up of 10 epochs, base learning rate of 0.15 and cosine annealing schedule using the LARS optimizer [65]. We also use the prediction head by default. All the ablations are performed using the ImageNet linear evaluation setting.

Nearest-neighbors as positives. Our core contribution in this paper is using nearest-neighbors (NN) as positives in the context of contrastive self-supervised learning. Here we investigate how this particular change, using nearest neighbors as positives, affects performance in various settings with and without momentum encoders. This analysis is presented in Table 4. First we show using the NNs in contrastive learning (row 2) is 3% better in Top-1 accuracy than using view 1 embeddings (similar to SimCLR) shown in row 1. We also explore using momentum encoder (similar to MoCo [33]) in our contrastive setting. Here using NNs also improves the top-1 performance by 2.4%.

Data Augmentation. Both SimCLR [11] and BYOL [28] rely heavily on a well designed data augmentation pipeline to get the best performance. However, NNCLR is less dependent on complex augmentations as nearest-neighbors already provide richness in sample variations. In this experiment, we remove all color augmentations and Gaussian blur, and train with random crops as the only method of augmentation for 300 epochs following the setup used in [28]. We present the results in Table 5. We notice NNCLR achieves 68.2% top-1 performance on the ImageNet linear evaluation task suffering a performance drop of only 4.7%. On the other hand, SimCLR and BYOL suffer larger relative drops in performance, 27.6% and 13.1% respectively. The performance drop reduces further as we train our approach longer. With 1000 pre-training epochs, NNCLR with all augmentations achieves 74.9% while with only random crops NNCLR manages to get 73.3%, further reducing the gap to just 1.6%. While NNCLR also benefits from complex data augmentation operations, the reliance on color jitter and blurring operations is much less. This is encouraging for adopting NNCLR for pre-training in domains where data transformations used for ImageNet might not be suitable.

Mom. Enc.	Positive	Top-1	Top-5
	View 1	71.4	90.4
	NN of View 1	**74.5**	**91.9**
✓	View 1	72.5	91.3
✓	NN of View 1	**74.9**	**92.1**

Table 4: **Effect of adding nearest-neighbors as positives** in various settings. Results are obtained for ImageNet linear evaluation.

Method	SimCLR [11]	BYOL [28]	NNCLR
Full aug.	67.9	72.5	72.9
Only crop	40.3 (↓ -27.6)	59.4 (↓ -13.1)	**68.2** (↓ -4.7)

Table 5: **Performance with only crop augmentation.** ImageNet top-1 performance for linear evaluation is reported.

Method	100	200	400	800
SimCLR [11]	66.5	68.3	69.8	70.4
MoCov2 [13]	67.4	69.9	71.0	72.2
BYOL [28]	66.5	70.6	73.2	74.3
SWAV [8]	66.5	69.1	70.7	71.8
SimSiam [14]	68.1	70.0	70.8	71.3
NNCLR	**69.4**	**70.7**	**74.2**	**74.9**

Table 6: **Number of pre-training epochs vs. performance.** Results are obtained for ImageNet linear evaluation.

Pre-training epochs. In Table 6, we show how our method compares to other methods when we have different pre-training epoch budgets. NNCLR is better than other self-supervised methods when pre-training budget is kept constant. We find that base learning rate of 0.4 works best for 100 epochs, and 0.3 works for 200, 400 and 800 epochs.

Support set size. Increasing the size of the support set increases performance in general. We present results of this experiment in Table 7a. By using a larger support set, we increase the chance of getting a closer nearest-neighbour from the full dataset. As also shown in Table 7b, getting the closest (i.e. top-1) nearest-neighbour obtains the best performance, even compared against top-2.

We also find that increasing the support set size beyond $98,304$ doesn't lead to any significant increase in performance possibly due to an increase in the number of stale embeddings in the support set.

Nearest-neighbor selection strategy. Instead of using the nearest-neighbor, we also experiment with taking one of the top-k NNs randomly. These results are presented in Table 7b. Here we investigate whether increasing the diversity of nearest-neighbors (i.e. increasing k) results in improved performance. Although our method is somewhat robust to changing the value of k, we find that increasing the top-k beyond $k = 1$ always results in slight degradation in performance. Inspired by recent work [21, 24] we also investigate using a soft-nearest neighbor, a convex combination of embeddings in the support-set where each one is weighted by its similarity to the embedding (see [21] for details). We present results in Table 7e. We find that the soft nearest neighbor can be used for training but results in worse performance than using the hard NN.

Batch size. Batch size has shown to be an important factor that affects performance, particularly in the contrastive learning setting. We vary the batch size and present the results in Table 7c. In general, larger batch sizes improve the performance peaking at 4096.

Embedding size. Our method is robust to choice of the embedding size as shown in Table 7d. We vary the embedding size in powers of 2 from 128 to 2048 and find similar performance over all settings.

Prediction head As shown in Table 7f, adding a prediction head results in a modest 0.4% boost in the top-1 performance.

Different implementations of support set. We also investigate some variants of how we can implement the support set from which we sample the nearest neighbor. We present results of this experiment in Table 8. In the first row, instead of using a queue, we pass a random set of images from the dataset through the current encoder and use the nearest neighbor from that set of embeddings. This works reasonably well but we are limited by how many examples we can fit in accelerator memory. Since we cannot increase the size of this set beyond 16384, which results in sub-optimal performance. Also using a random set of size 16384 is about four times slower than using a queue (when training with a batch size of 4096) as each forward pass requires four times more samples through the encoder. This experiment also shows that NNCLR does not need features of past samples (akin to momentum encoders [33]) to learn representations. We also experiment with updating the elements in the support set randomly as opposed to the default FIFO manner. We find that FIFO results in more than 2% better ImageNet linear evaluation Top-1 accuracy.

Compute overhead. We find increasing the size of the queue results in improved performance but this improvement comes at a cost of additional memory and compute required during training. In Table 9 we show how queue scaling with $d = 256$ affects memory required during training and number of training steps per second. With a support size of about $98k$ elements we require a modest 100 MB more in memory.

4.5. Discussion

Ground Truth Nearest Neighbor. We investigate two aspects of the NN: first, how often does the NN have the same ImageNet label as the query; and second, if the NN is always picked to be from the same ImageNet class (with an Oracle algorithm), then what is the effect on training and the final performance? Figure 3, shows how the accuracy of the NN picked from the queue varies as training proceeds. We observe that towards the end of training the accuracy of picking the right neighbor (i.e. from the same class) is about 57%. The reason that it is not higher is possibly due to random crops being of the background, and thus not containing the object described by the ImageNet label.

We next investigate if NNCLR can achieve better performance if our top-1 NN is always from the same ImageNet class. This is quite close to the supervised learning setup except instead of training to predict classes directly, we train using our self-supervised setup. A similar experiment has also been described as *UberNCE* in [30]. This experiment verifies if our training dynamics prevent the model from converging to the performance of a supervised learning baseline even when the true NN is known. To do so, we store the ImageNet labels of each element in the queue and always pick a NN with the same ImageNet label as the query view. We observe that with such a setting we achieve 75.8% accuracy in 300 epochs. With the Top-1 NN from the support set, we manage to get 72.9% in 300 epochs. This suggests that there is still a possibility of improving performance with a better NN picking strategy, although it might be hard to design one that works in a purely unsupervised way.

Training curves. In Figure 4 we show direct comparison between training using cross-entropy loss with an augmen-

Queue Size	8192	16384	32768	65536	98304
Top-1	73.6	74.2	74.9	75.0	**75.4**
Top-5	91.2	91.7	92.1	92.2	**92.3**

(a) Support set size

k in k-NN	1	2	4	8	16	32
Top-1	**74.9**	74.1	73.8	73.8	73.8	73.2
Top-5	**92.1**	91.6	91.5	91.4	91.3	91.2

(b) Varying k in k-NN

Batch size	256	512	1024	2048	4096	8192
Top-1	68.7	71.7	72.9	73.5	**74.9**	74.3
Top-5	88.7	90.4	91.1	91.6	**92.1**	91.9

(c) Batch size.

d	128	256	512	1024	2048
Top-1	74.9	74.9	74.8	74.9	74.6
Top-5	92.1	92.1	92.0	92.0	92.0

(d) Varying embedding size d

Type of NN	Top-1	Top-5
Soft nearest-neighbor	71.4	90.4
Hard nearest-neighbor	**74.9**	**92.1**

(e) Soft vs. hard nearest neighbors as positives.

Prediction MLP	Top-1	Top-5
	74.5	92.0
✓	**74.9**	**92.1**

(f) Effect of prediction head.

Table 7: **NNCLR Ablation Experiments.** Results are obtained for ImageNet linear evaluation.

Support set variant	Size	Top-1	Top-5
NNs from current encoder	16384	74.0	91.8
NNs from queue (older embeddings)	16384	74.2	91.7

Table 8: **Different implementations of the support set.**

Queue Size	8192	16384	32768	65536	98304
Memory (MB)	8.4	16.8	33.6	67.3	100.8
Steps per sec	6.41	6.33	6.14	5.95	5.68

Table 9: **Queue size computational overheads.**

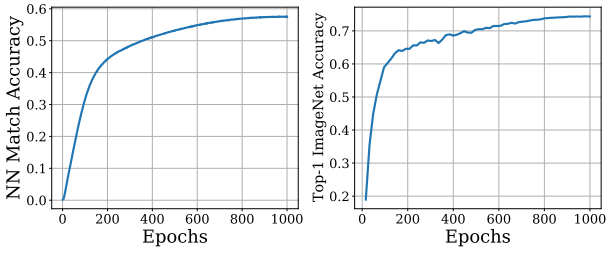


Figure 3: **NN Match Accuracy vs. Performance.**

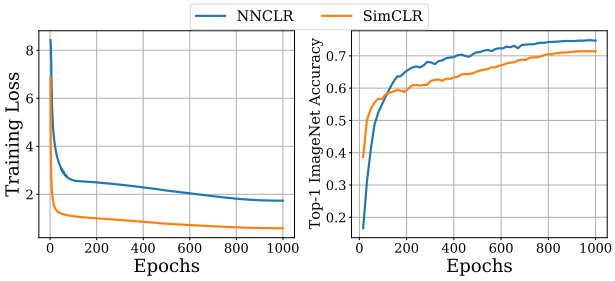


Figure 4: **NNCLR vs SimCLR** Training curves and linear evaluation curves.

tation of the same view as positive (SimCLR) and training with NN as positive (NNCLR). The training loss curves indicate NNCLR is a more difficult task as the training needs to learn from hard positives from other samples in the dataset. Linear evaluation on ImageNet classification shows that it takes about 120 epochs for NNCLR to start outperforming SimCLR, and remains higher until the end of pre-traininig at 1000 epochs.

NNs in Support Set. In Figure 5 we show a typical batch of nearest neighbors retrieved from the support set towards the end of training. Column 1 shows examples of view 1, while the other elements in each row shows the retrieved nearest neighbor from the support set. We hypothesize that the improvement in performance is due to this diversity introduced in the positives, something that is not covered by pre-defined data augmentation schemes. We observe that while many times the retrieved NNs are from the same class, it is not uncommon for the retrieval to be based on other similarities like texture. For example, in row 3 we observe retrieved images are all of underwater images and row 4 retrievals are not from a dog class but are all images with cages in them.

5. Conclusion

We present an approach to increase the diversity of positives in contrastive self-supervised learning. We do so by using the nearest-neighbors from a support set as positives. NNCLR achieves state of the art performance on multiple datasets. Our method also reduces the reliance on data augmentation techniques drastically.

Figure 5: **Nearest neighbors from support set** show the increased diversity of positives in NNCLR.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019. 2

[2] Mehdi Azabou, Mohammad Gheshlaghi Azar, Ran Liu, Chi-Heng Lin, Erik C Johnson, Kiran Bhaskaran-Nair, Max Dabagia, Keith B Hengen, William Gray-Roncal, Michal Valko, et al. Mine your own view: Self-supervised learning through across-sample prediction. *arXiv preprint arXiv:2102.10106*, 2021. 3

[3] Miguel A Bautista, Artsiom Sanakoyeu, Ekaterina Sutter, and Björn Ommer. Cliquecnn: Deep unsupervised exemplar learning. *arXiv preprint arXiv:1608.08792*, 2016. 2

[4] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2014. 5

[5] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 5

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 2

[7] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2959–2968, 2019. 2

[8] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 2, 3, 5, 6

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 14, 15

[10] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. 2010. 2

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1, 2, 3, 4, 5, 6

[12] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 2, 4, 5

[13] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2, 5, 6

[14] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020. 5, 6, 11, 13

[15] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 3

[16] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5

[17] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 2

[18] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012. 3

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 13

[20] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. Citeseer, 2014. 2

[21] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1801–1810, 2019. 3, 7

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 5

[23] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incre-

mental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 2004. 5

[24] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In *International Conference on Machine Learning*, pages 2012–2020. PMLR, 2019. 7

[25] Dedre Gentner and Laura L Namy. Comparison in the development of categories. *Cognitive Development*, 14(4):487–513, 1999. 1

[26] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6928–6938, 2020. 2

[27] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 2

[28] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 1, 2, 3, 4, 5, 6, 13

[29] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006. 2

[30] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *NeurIPS*, 2020. 2, 3, 7

[31] James Hays and Alexei A Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (ToG)*, 26(3):4–es, 2007. 3

[32] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In *2008 ieee conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008. 3

[33] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 1, 2, 3, 4, 6, 7

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 13

[35] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020. 5

[36] Jiabo Huang, Qi Dong, Shaogang Gong, and Xiatian Zhu. Unsupervised deep learning by neighbourhood discovery. In *International Conference on Machine Learning*, pages 2849–2858. PMLR, 2019. 2

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[38] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021. 4

[39] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020. 2

[40] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 491–507. Springer, 2020. 13

[41] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 5

[42] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[43] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020. 5

[44] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 5

[45] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020. 5

[46] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 5

[47] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 3

[48] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 5

[49] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017. 2

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 4

[51] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 2

[52] Saurabh Singh, Abhinav Gupta, and Alexei A Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, pages 73–86.

Springer, 2012. 3

[53] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865, 2016. 3

[54] James Thewlis, Samuel Albanie, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of landmarks by descriptor vector exchange. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6361–6371, 2019. 3

[55] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2, 5

[56] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020. 5

[57] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 13, 14

[58] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015. 2

[59] Zhirong Wu, Alexei A Efros, and Stella X Yu. Improving generalization via scalable neighborhood component analysis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 685–701, 2018. 2, 3

[60] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 1, 2, 3, 5

[61] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, June 2010. 5

[62] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016. 2

[63] Xueting Yan, Ishan Misra, Abhinav Gupta, Deepti Ghadiyaram, and Dhruv Mahajan. Clusterfit: Improving generalization of visual representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6509–6518, 2020. 2

[64] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5147–5156, 2016. 2

[65] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 5, 6

[66] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6002–6012, 2019. 2, 3

Appendix

A. Pseudo-code

In Algorithm 1 we present the pseudo-code of NNCLR.

Algorithm 1 Pseudocode

```
# f: backbone encoder + projection MLP
# g: prediction MLP
# Q: queue

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = g(z1), g(z2) # predictions, n-by-d

   NN1 = NN(z1, Q) # top-1 NN lookup, n-by-d
   NN2 = NN(z2, Q) # top-1 NN lookup, n-by-d

   loss = L(NN1, p2)/2 + L(NN2, p1)/2

   loss.backward() # back-propagate
   update(f, g) # SGD update
   update_queue(Q, z1) # Update queue with latest
       projection embeddings

def L(nn, p, temperature=0.1):
   nn = normalize(nn, dim=1) # l2-normalize
   p = normalize(p, dim=1) # l2-normalize

   logits = nn @ p.T # Matrix multiplication, n-by-n
   logits /= temperature # Scale by temperature

   n = p.shape[0] # mini-batch size
   labels = range(n)

   loss = cross_entropy(logits, labels)

   return loss

def NN(z, Q):
  z = normalize(z, dim=1) # l2-normalize
  Q = normalize(Q, dim=1) # l2-normalize
  sims = z @ Q.T
  nn_idxes = sims.argmax(dim=1) # Top-1 NN indices
  return Q[nn_idxes]
```

It is possible to use momentum encoder with NNCLR training. The pseudo-code when momentum encoder is used is shown in Algorithm 2.

B. Evolution of Nearest-Neighbors

In Figure 6 we show how the nearest-neighbors (NN) vary as training proceeds. We observe consistently that in the beginning of training the NNs are usually chosen on the basis of color and texture. As the encoder becomes better at recognizing classes later in training, the NNs tend to belong to similar semantic classes.

C. SimSiam with Nearest-neighbor as Positive

In this experiment we want to check if it is possible to use the nearest-neighbor in a non-contrastive loss. To do so we use the self-supervised framework SimSiam [14], in which the authors use a mean squared error on the embeddings of

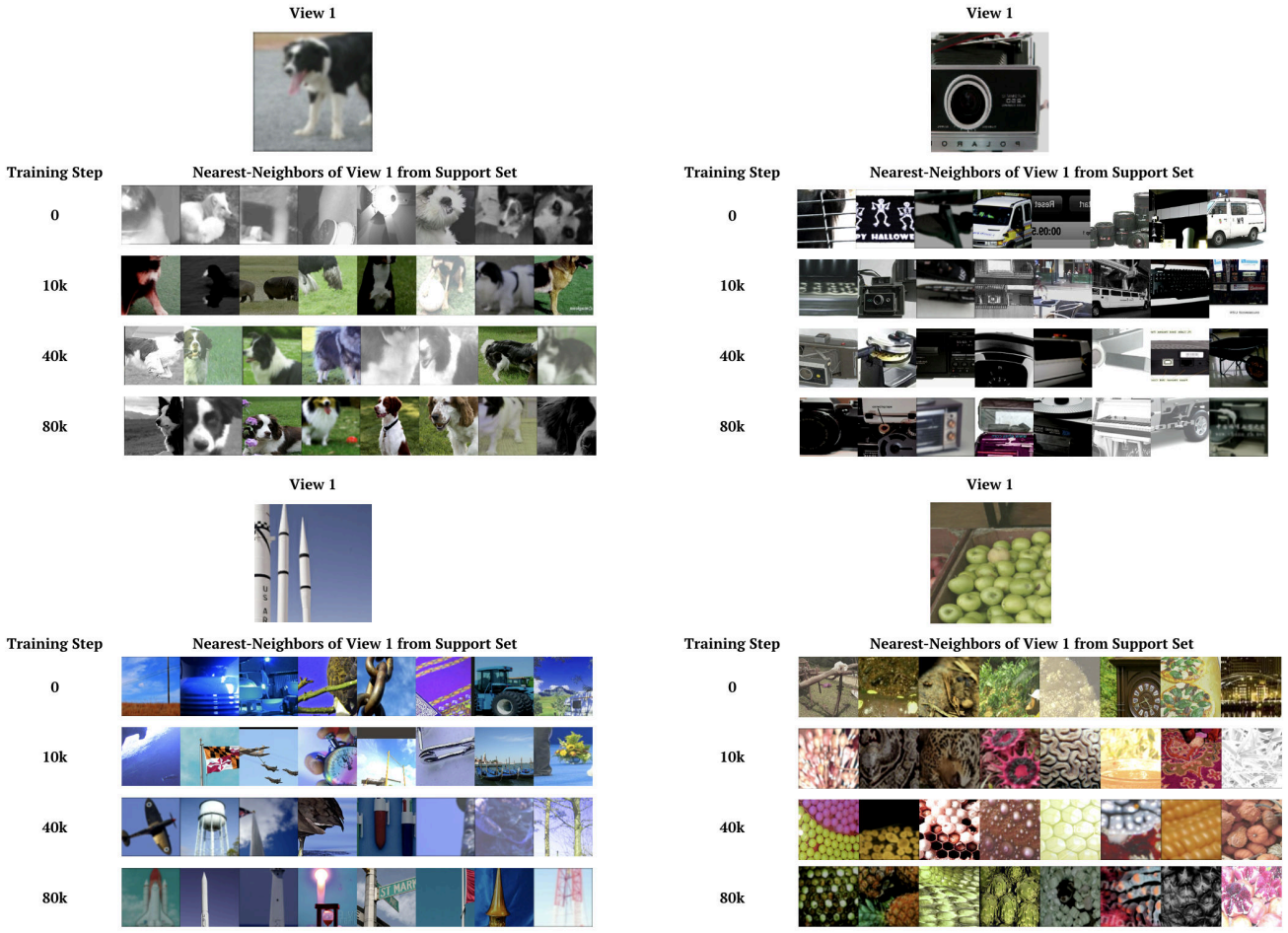
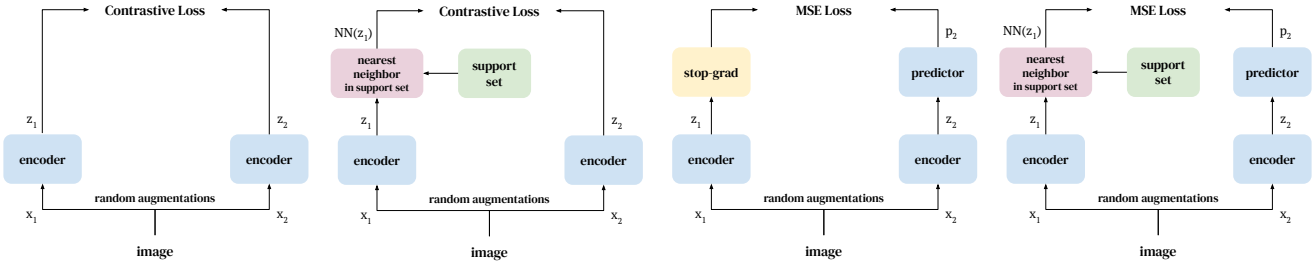


Figure 6: **Evolution of Nearest-neighbors** as training proceeds.



(a) SimCLR (b) NNCLR (c) SimSiam (d) NNSiam

Figure 7: **Comparison of self-supervised methods.**

the 2 views, where one of the branches has a stop-gradient and the other one has a prediction MLP. We replace the stop-gradient branch with its nearest-neighbor from the support set. We call this method NNSiam. In Figure 7 we show how NNSiam differs from SimSiam. We also show how they both differ from SimCLR and NNCLR. Note that there is an implicit stop-gradient in NNSiam because of the use of hard nearest-neighbors. For this experiment, we use an embedding size of 2048, which is the same dimensionality used in SimSiam. We train with a batch size of 4096 with the LARS optimizer with a base learning rate of 0.2. We find that even with the non-contrastive loss using the nearest neighbor as positive leads to 1.3% improvement in accuracy under ImageNet linear evaluation protocol. This shows that the idea of using harder positives can lead to performance improvements outside the InfoNCE loss also.

Algorithm 2 Pseudocode with Momentum Encoder

```
# f: backbone encoder + projection MLP
# f_m: momentum version of (backbone encoder +
    projection MLP)
# g: prediction MLP
# Q: queue
# t: tau for momentum encoder

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = g(z1), g(z2) # predictions, n-by-d

   zm1, zm2 = f_m(x1), f_m(x2) # projections, n-by-d

   NN1 = NN(zm1, Q) # top-1 NN lookup, n-by-d
   NN2 = NN(zm2, Q) # top-1 NN lookup, n-by-d

   loss = L(NN1, p2)/2 + L(NN2, p1)/2

   loss.backward() # back-propagate
   update(f, g) # SGD update
   update_queue(Q, z_m1) # Update queue with latest
        projection embeddings from momentum encoder

   f_m = t*f_m + (1 - t) * f # Update momentum
        encoder weights
```

Method	Positive	Top-1	Top-5
SimSiam[14]	View 1	71.3	-
NNSiam	NN of View 1	72.6	90.4

Table 10: **SimSiam with nearest-neighbor as positive.**

D. Experiments with Vision Transformers

Vision Transformers (ViT) [19] are a class of architectures introduced recently to process images using Transformers. We explore the effectiveness of using self-supervised learning methods to train vision transformers. We find the Adam optimizer to be effective for training ViT models. We train for 1000 epochs using 2 crops with a base learning rate of 3×10^{-4} with a warmup schedule of 10 epochs followed by cosine decay learning schedule, weight decay of 0.05 and a stochastic depth dropout of 0.1. We use the output corresponding to the [CLS] token of the final layer as the embedding z_i in the loss and as the representation used as input for the linear classifier. We present the results of this experiment in Table 11. In our experiments, we find NNCLR well suited to train Visual Transformers outperforming the supervised learning model (trained without the augmentation introduced in DeIT [57]) by 4.4% and the SimCLR model by 2%.

Arch.	Training Loss	Top-1
ViT-B/16	Supervised Learning	72.1
ViT-B/16 [57]	Supervised Learning	**81.8**
ViT-B/16	SimCLR	74.5
ViT-B/16	NNCLR	**76.5**

Table 11: **Vision Transformer** Top-1 ImageNet accuracy under the linear protocol.

E. Self-supervised Learning as a Pre-training Step for Supervised Learning

Until recently it was believed supervised learning on a particular dataset would always be better than self-supervised learning on that dataset. BYOL [28] showed that some large models can end up achieving higher accuracy than their supervised counterparts. In this experiment, we similarly show that self-supervised learning can serve as a useful pre-training step for supervised learning, alleviating the need for extra data or complex regularization techniques that are used to boost the performance of the final model.

In this experiment we first pretrain a model with NNCLR for 1000 epochs on the ImageNet 2012 dataset. We then proceed to perform regular supervised training for 100 epochs. This is similar to the semi-supervised learning setup but with 100% of the labels available for fine-tuning the pre-trained model. In Table 12 we show results on this experiment. We observe initialization from a self-supervised model improves the performance of ResNet50 from 76.2% to 79.1%. This performance is comparable to training a ResNet-50 model with JFT-300M dataset [40] which is considerably more data than ImageNet. We find this pre-training technique is especially useful with the newly proposed ViT [19] architecture which requires additional data or regularization techniques to achieve good performance. ViT-B/16 pre-trained with NNCLR outperforms DeIT [57] but is only worse than the same architecture trained with the JFT-300M dataset by 0.5%. This experiment highlights another use-case for self-supervised learning: to provide a good initialization for supervised learning.

Arch.	Res.	SS PT	Reg. / Extra Data	Top-1
ResNet50 [34]	224	-	-	76.2
ResNet50 [40]	224	-	JFT-300M	79.0
ResNet50	224	NNCLR	-	**79.1**
ViT-S/16	224	-	DeIT Aug.	79.8
ViT-S/16	224	NNCLR	DeIT Aug.	**82.7**
ViT-B/16 [57]	224	-	DeIT Aug.	81.8
ViT-B/16	224	NNCLR	DeIT Aug.	**82.5**
ViT-B/16 [19]	384	-	-	77.9
ViT-B/16 [57]	384	-	DeIT Aug.	83.1
ViT-B/16 [19]	384	-	JFT-300M	**84.2**
ViT-B/16	384	NNCLR	DeIT Aug.	83.7
ViT-B/8	224	-	DeIT Aug.	83.1
ViT-B/8	224	NNCLR	DeIT Aug.	**84.4**

Table 12: **Self-supervised learning as pre-training for supervised learning.** Top-1 ImageNet accuracy. Both self-supervised and supervised training are done on ImageNet 2012 training set. Arch: Architecture, SS PT: Self-supervised Pre-training Technique, Reg. Additional regularization techniques

F. Transfer Learning

In this section we study the performance of Visual Transformers (ViT) in the transfer learning setting. To do this, we repeat the experiment described in Section 4.3 with ViT models. The results of this experiment are presented in Table 13. First, we observe that ViT-B/16 trained with only a supervised learning objective using the data augmentation strategy outlined in DeIT[57] on the ImageNet dataset transfers poorly as compared to ResNet50 architecture trained with the supervised loss. ViT-B/16 is worse on 10 datasets out of the 12 datasets in the transfer learning benchmark. However, ViT-B/16 trained with NNCLR objective outperforms the ResNet50 architecture trained with the same loss on 8 out of the 12 datasets in the benchmark. Additionally, ViT-B/16 trained with NNCLR has better performance on all datasets as compared to the same model trained with just the supervised loss. This shows that the Vision Transformer's potential for transfer learning is enhanced by using self-supervised training like NNCLR. Of particular note is the increase in performance on Birdsnap ($\sim$ 10.7%), Sun397 ($\sim$ 5.8%), Cars ($\sim$ 8.0%), Aircraft ($\sim$ 13.2%), DTD ($\sim$ 5.8%) and Flowers ($\sim$ 9.3%). We also train a ViT-B/8 model that has the same architecture as ViT-B/16 but uses 8×8-sized non-overlapping patches in the images to produce the tokens used in the Transformer architecture. We find that ViT-B/8 outperforms ViT-B/16 on all datasets in the transfer learning setup. We also observe that NNCLR brings significant gains over just supervised learning. Of particular note is the increase in performance on Birdsnap ($\sim$ 9.8%), Sun397 ($\sim$ 3.9%), Aircraft ($\sim$ 10.5%), DTD ($\sim$ 7.1%) and Flowers ($\sim$ 5.7%). Finally, we test the transfer learning performance on models pre-trained with NNCLR and fine-tuned with the supervised loss as outlined in Section E. We find this setup increases the performance over the already strong baseline of ViT-B/8 trained with just NNCLR. We find a boost of 8.3% on Food, 5.3% on Birdsnap, 2.9% on Sun397, 15.8% on Cars, 2.2% on VOC2007, 2.1% on Pets. However, we also observe fine-tuning on the supervised loss also results in degradation in performance on some datasets: 1.1% on DTD and 1.1% on Flowers. Inspite of the degradation in performance, ViT-B/8 pretrained with NNCLR and fine-tuned with the supervised loss on ImageNet is always better than just using the supervised loss only. Overall, we find Vision Transformers are well suited for transfer learning if they have been pre-trained with a self-supervised loss like NNCLR.

G. Visualizations

G.1. Attention in NNCLR ResNet

We visualize "attention" of various models trained with NNCLR to probe what the model might be focusing on. In order to visualize attention, we need a query embedding and the feature map produced by the image. We convolve the query embedding across the feature map to produce an attention map. In Figure 8, we show examples of attention corresponding to different query embeddings on images from the COCO dataset. We use the original resolution of the images to get a larger feature map that can highlight object boundaries and locations of small objects. Each arrow points to the query embedding in the image and the corresponding attention of that query embedding in the feature map. In each sub-figure's caption we also mention the object class they are pointing at. In [9], the authors observe that vision transformer models trained with self-supervised losses like DINO [9] show emergence of properties like objectness and part localization in the attention layer of the transformer. We find these emerging properties also exist in ResNet50 models trained with self-supervised losses like NNCLR.

G.2. Cross-image Attention with NNCLR ResNets

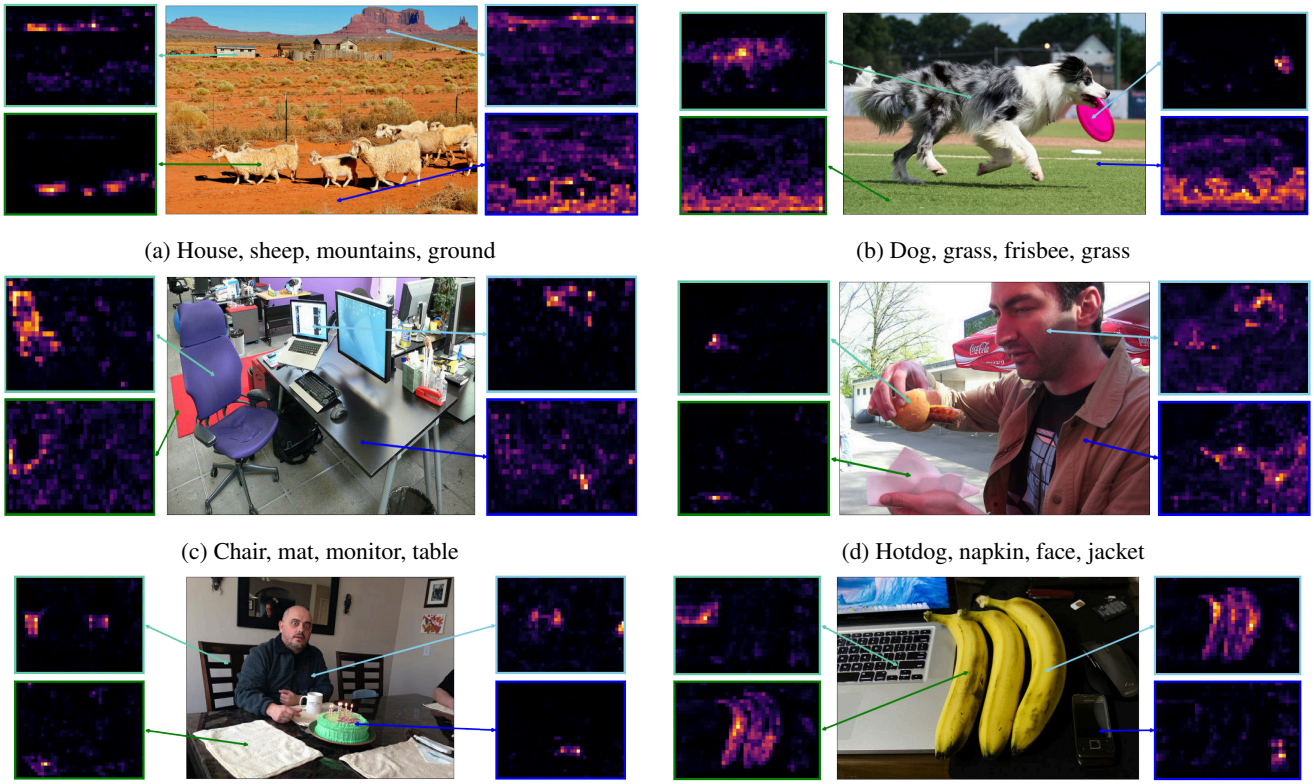
In the previous section we presented attention of a query embedding with parts of the same image. To visualize if models trained with NNCLR encode object categories, we conduct the following experiment. We use a query embedding of an object from one image by average pooling the features in the bounding box of an object denoted by the red box. We convolve this query embedding over the feature maps obtained by passing other images through a ResNet50 trained with NNCLR loss. We show the results of this experiment in Figure 9. We observe that features of the same object in different images are close to each other in the NNCLR feature space. The results of this experiment highlight that the learned features encode semantic similarity beyond color, in order to localize objects of the same category across different images as shown in Figure 9. In the first row, the features are able to localize multiple objects of the same category *giraffe*. In the second row, we show that while the ground-truth class is that of *stop-sign* the features are considering different kinds of street signs (hotel name, street name sign, railroad-crossing sign) as similar. This is interesting because the query features are pooled only from the stop-sign region but the retrieved regions of similarity are of different colors. We also observe the model does not activate over the text of the stop-sign. In the last two rows, we show how the model is able to localize small objects (like ball and frisbee) in different images.

G.3. Comparison of Attention in ResNets vs ViT

In this experiment we want to compare the attention maps of 2 architectures, ResNet-50 and ViT-B/16, each with 3 different weights: randomly initialized, ImageNet supervised and ImageNet NNCLR. For ViT we use the average self-attention over all heads in the final layer for the [CLS] token. For ResNet-50 we use the method described

Method	Arch.	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Cal.101	Flowers
BYOL	R50	75.3	91.3	78.4	57.2	62.2	67.8	60.6	82.5	75.5	90.4	94.2	96.1
SimCLR	R50	72.8	90.5	74.4	42.4	60.6	49.3	49.8	81.4	75.7	84.6	89.3	92.6
Sup.-IN	R50	72.3	93.6	78.3	53.7	61.9	66.7	61.0	82.8	74.9	91.5	**94.5**	94.7
NNCLR	R50	76.7	93.7	79.0	61.4	62.5	67.1	64.1	83.0	75.5	91.8	91.3	95.1
Sup. IN	ViT-B/16	71.4	95.2	80.5	52.8	59.3	50.7	52.1	81.2	71.5	91.4	85.6	85.6
NNCLR	ViT-B/16	72.7	95.7	82.3	63.5	65.5	58.3	65.3	83.2	77.3	92.5	88.5	94.9
Sup. IN	ViT-B/8	76.2	95.9	81.4	61.9	62.6	61.4	58.6	83.8	72.1	93.2	85.5	90.1
NNCLR	ViT-B/8	77.2	**96.8**	82.0	71.7	66.5	58.6	**69.1**	84.2	**79.2**	92.8	88.7	**95.8**
NNCLR + Sup	ViT-B/8	**85.5**	96.2	**84.7**	**77.0**	**69.4**	**74.4**	68.8	**86.4**	78.1	**94.9**	90.5	94.7

Table 13: **Transfer learning performance** using ResNet-50 and ViT-B/16 pretrained with ImageNet. For all datasets we report Top-1 classification accuracy except Aircraft, Caltech-101, Pets and Flowers for which we report mean per-class accuracy and VOC2007 for which we report 11-point MAP.



(a) House, sheep, mountains, ground

(b) Dog, grass, frisbee, grass

(c) Chair, mat, monitor, table

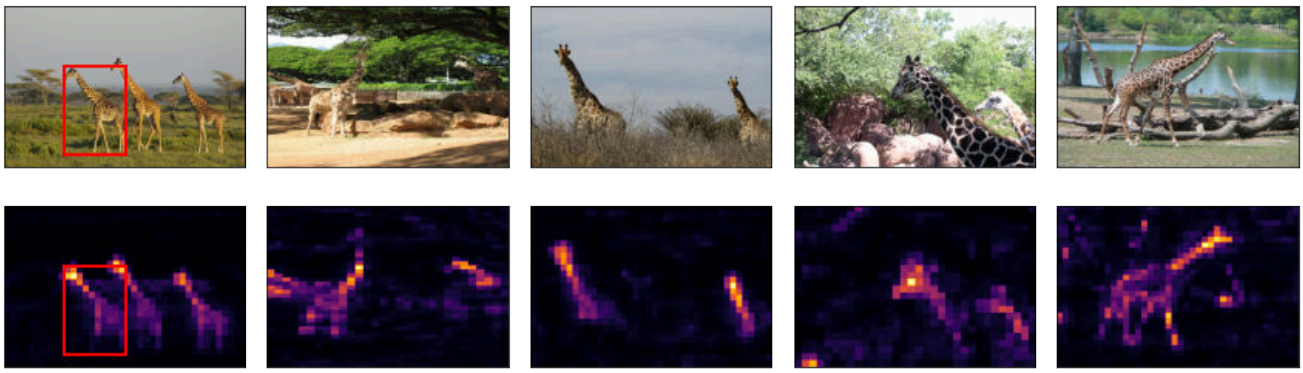
(d) Hotdog, napkin, face, jacket

(e) Chair, napkin, person, cake

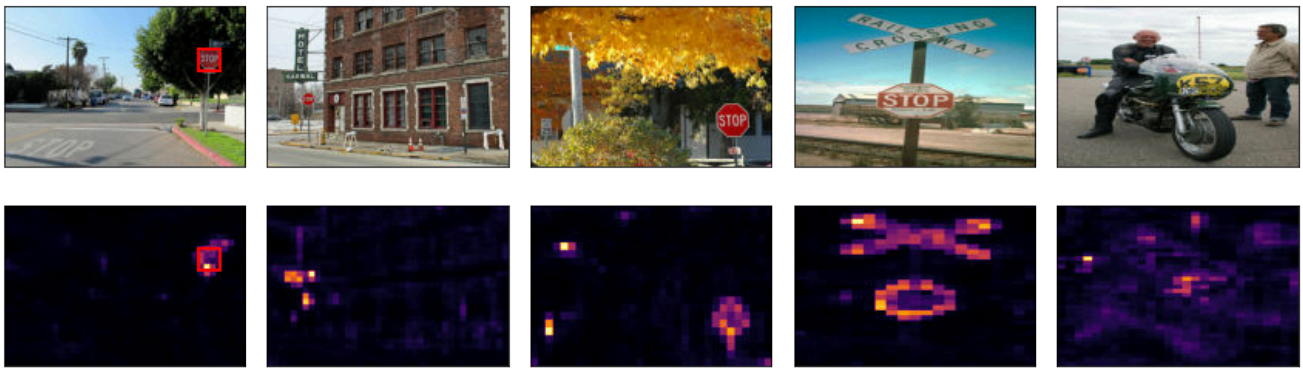
(f) Keyboard, banana, banana, mobile phone

Figure 8: **Visualization of Attention in NNCLR ResNets.** We observe that self-supervised pre-training with NNCLR results in localization of objects of different categories.

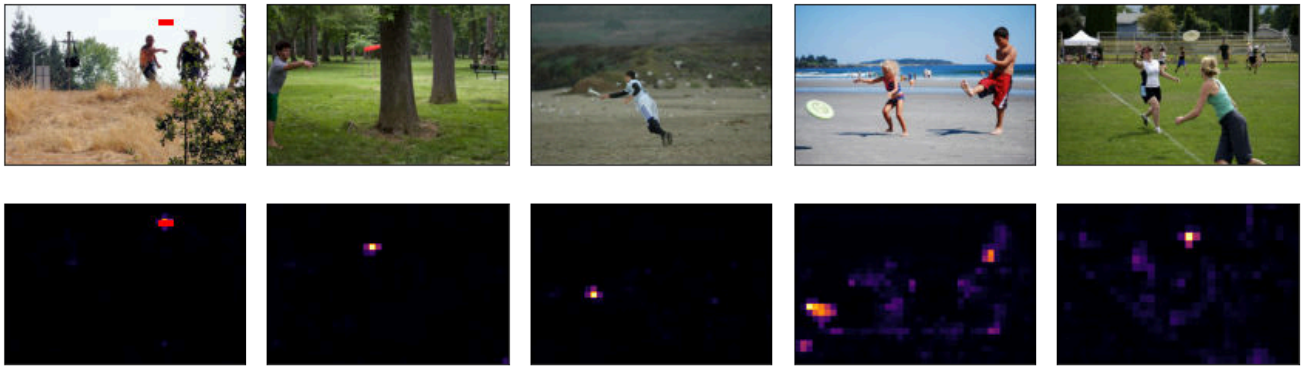
in Sec. G.1 and use the average pooled embedding as the query embedding since it is the equivalent of the [CLS] token in ViTs. We show our results in Figure 10. First we observe the attention maps delineate salient objects in the image. Similar to DINO [9] we observe that ViTs trained with just a supervised learning objective do not have objects highlighted in their attention map. However, we do observe that both supervised and self-supervised ResNets have delineated objects in their attention maps. We also note that sometimes randomly initialized ResNets (rows 2 and 5) and ViTs (rows 4 and 7) are able to localize individual objects in their attention map because of similarity in color and texture across the spatial extent of the object. This fact makes it difficult to conclude that a model that produces the well-delineated objects in the self-attention map will necessarily have learned semantically meaningful features. We suggest using a combination of self-attention and cross-attention maps (described in Section G.2) as a more robust visualization technique for interpreting semantic features.



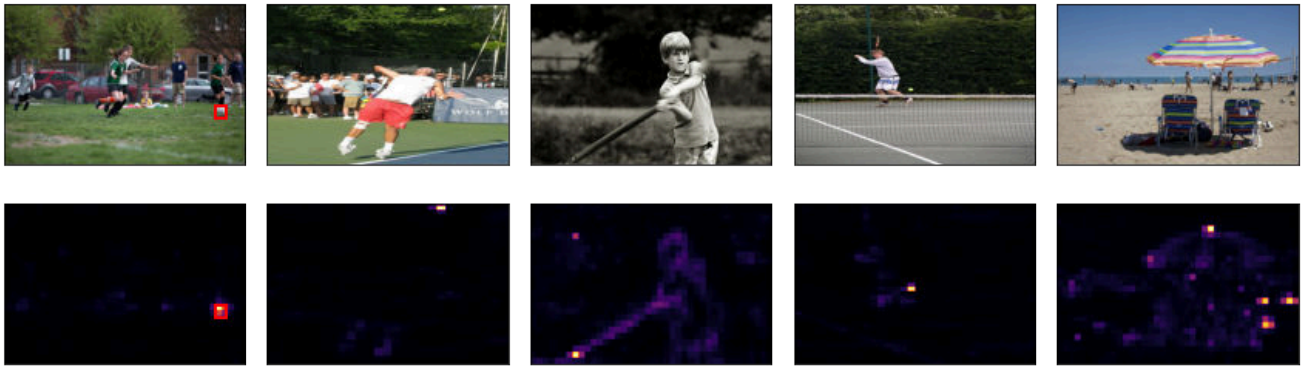
(a) Giraffe



(b) Stop-sign



(c) Frisbee



(d) Ball

Figure 9: **Cross-Attention.** We use features of an object in the images in the first column (red box in leftmost column shows the query object) to look for the same object in images in other columns. We find NNCLR features of the same object in different images are close to each other.

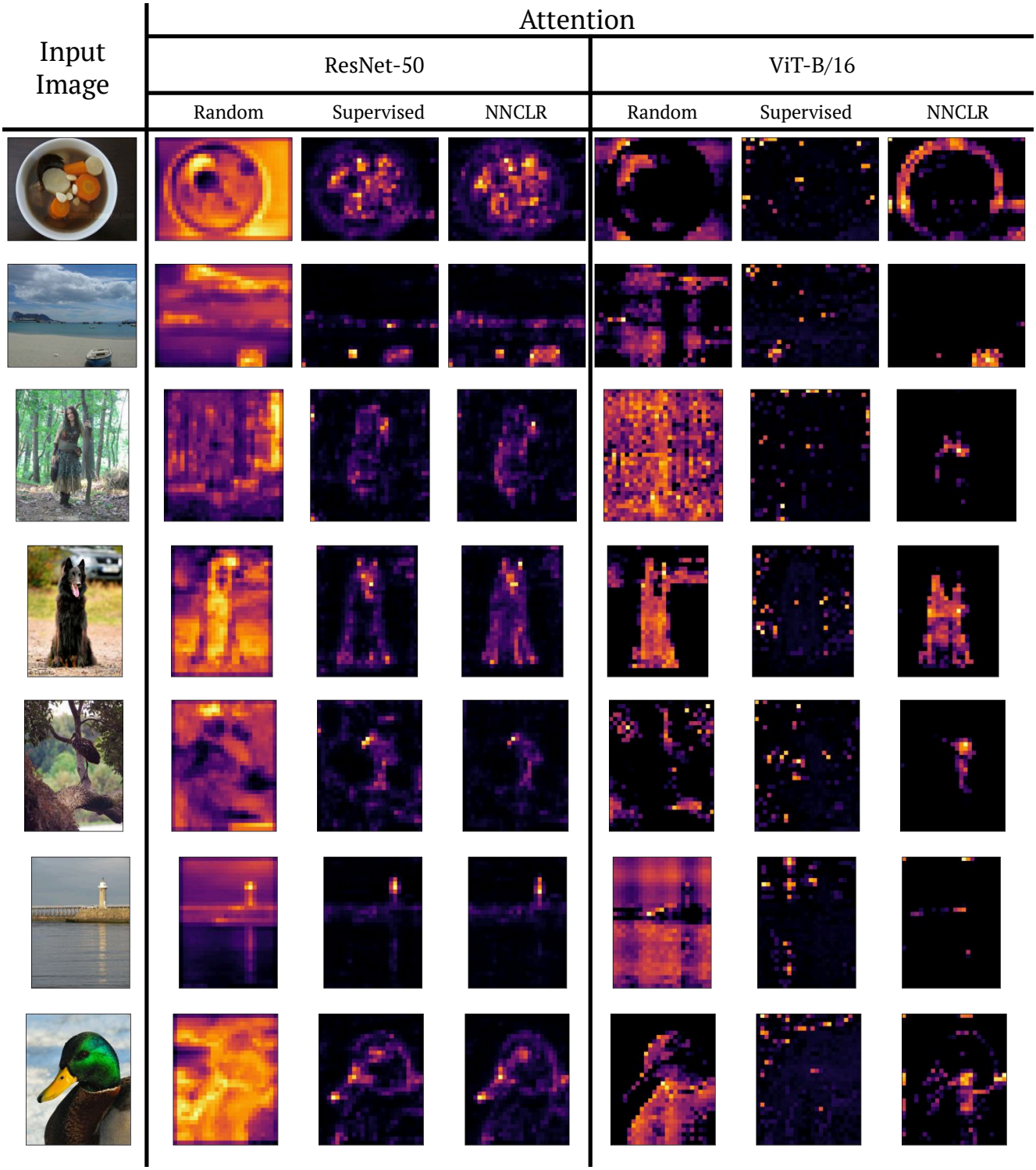


Figure 10: **Comparison of Attention Maps.** We compare attention between feature map and the global image embedding of the last layer of ResNet-50 and ViT-B/16 architectures with three different sets of weights, *Random*: randomly initialized weights, *Supervised*: weights after training a model with supervised learning loss, *Supervised*: weights after training a model with the NNCLR objective. More details in Section G.3

.