Hard Negative Mixing for Contrastive Learning

Yannis Kalantidis **Mert Bulent Sariyildiz** **Noe Pion**

Philippe Weinzaepfel **Diane Larlus**

NAVER LABS Europe
Grenoble, France

Abstract

Contrastive learning has become a key component of self-supervised learning approaches for computer vision. By learning to embed two augmented versions of the same image close to each other and to push the embeddings of different images apart, one can train highly transferable visual representations. As revealed by recent studies, heavy data augmentation and large sets of negatives are both crucial in learning such representations. At the same time, data mixing strategies, either at the image or the feature level, improve both supervised and semi-supervised learning by synthesizing novel examples, forcing networks to learn more robust features. In this paper, we argue that an important aspect of contrastive learning, *i.e.* the effect of *hard negatives*, has so far been neglected. To get more meaningful negative samples, current top contrastive self-supervised learning approaches either substantially increase the batch sizes, or keep very large memory banks; increasing memory requirements, however, leads to diminishing returns in terms of performance. We therefore start by delving deeper into a top-performing framework and show evidence that harder negatives are needed to facilitate better and faster learning. Based on these observations, and motivated by the success of data mixing, we propose *hard negative mixing* strategies at the feature level, that can be computed on-the-fly with a minimal computational overhead. We exhaustively ablate our approach on linear classification, object detection, and instance segmentation and show that employing our hard negative mixing procedure improves the quality of visual representations learned by a state-of-the-art self-supervised learning method.

Project page: https://europe.naverlabs.com/mochi

1 Introduction

Contrastive learning was recently shown to be a highly effective way of learning visual representations in a self-supervised manner [12, 32]. Pushing the embeddings of two transformed versions of the same image (forming the positive pair) close to each other and further apart from the embedding of any other image (negatives) using a contrastive loss, leads to powerful and transferable representations. A number of recent studies [15, 27, 70] show that carefully handcrafting the set of data augmentations applied to images is instrumental in learning such represen-

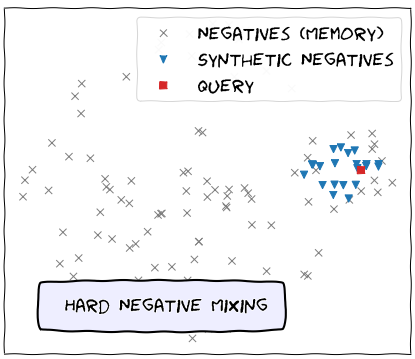


Figure 1: **MoCHi** generates synthetic hard negatives for each positive (query).

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

tations. We suspect that the right set of transformations provides more *diverse*, *i.e*. more challenging, copies of the same image to the model and makes the self-supervised (*proxy*) task harder. At the same time, data mixing techniques operating at either the pixel [76, 91, 93] or the feature level [75] help models learn more robust features that improve both supervised and semi-supervised learning on subsequent (*target*) tasks.

In most recent contrastive self-supervised learning approaches, the negative samples come from either the current batch or a memory bank. Because the number of negatives directly affects the contrastive loss, current top contrastive approaches either substantially increase the batch size [12], or keep large memory banks. Approaches like [51, 84] use memories that contain the whole training set, while the recent Momentum Contrast (or MoCo) approach of He et al. [32] keeps a queue with features of the last few batches as memory. The MoCo approach with the modifications presented in [15] (named MoCo-v2) currently holds the state-of-the-art performance on a number of target tasks used to evaluate the quality of visual representations learned in an unsupervised way. It is however shown [12, 32] that increasing the memory/batch size leads to diminishing returns in terms of performance: more negative samples does not necessarily mean *hard* negative samples.

In this paper, we argue that an important aspect of contrastive learning, *i.e*. the effect of hard negatives, has so far been neglected in the context of self-supervised representation learning. We delve deeper into learning with a momentum encoder [32] and show evidence that harder negatives are required to facilitate better and faster learning. Based on these observations, and motivated by the success of data mixing approaches, we propose *hard negative mixing*, *i.e*. feature-level mixing for hard negative samples, that can be computed on-the-fly with a minimal computational overhead. We refer to the proposed approach as **MoCHi**, that stands for "(**M**)ixing (**o**)f (**C**)ontrastive (**H**)ard negat(**i**)ves".

A toy example of the proposed hard negative mixing strategy is presented in Figure 1; it shows a t-SNE [48] plot after running MoCHi on 32-dimensional random embeddings on the unit hypersphere. We see that for each positive query (red square), the memory (gray marks) contains many easy negatives and few hard ones, *i.e*. many of the negatives are too far to contribute to the contrastive loss. We propose to mix only the hardest negatives (based on their similarity to the query) and synthesize new, hopefully also hard but more diverse, negative points (blue triangles).

Contributions. **a)** We delve deeper into a top-performing contrastive self-supervised learning method [32] and observe the need for harder negatives; **b)** We propose hard negative mixing, *i.e*. to synthesize hard negatives directly in the embedding space, on-the-fly, and adapted to each positive query. We propose to both mix pairs of the hardest existing negatives, as well as mixing the hardest negatives *with* the query itself; **c)** We exhaustively ablate our approach and show that employing hard negative mixing improves both the generalization of the visual representations learned (measured via their transfer learning performance), as well as the utilization of the embedding space, for a wide range of hyperparameters; **d)** We report competitive results for linear classification, object detection and instance segmentation, and further show that our gains over a state-of-the-art method are higher when pre-training for fewer epochs, *i.e*. MoCHi learns transferable representations faster.

2 Related work

Most early self-supervised learning methods are based on devising proxy classification tasks that try to predict the properties of a transformation (*e.g*. rotations, orderings, relative positions or channels) applied on a single image [18, 19, 25, 40, 53]. Instance discrimination [84] and CPC [54] were among the first papers to use contrastive losses for self-supervised learning. The last few months have witnessed a surge of successful approaches that also use contrastive learning losses. These include MoCo [15, 32], SimCLR [12, 13], PIRL [51], CMC [69] or SvAV [11]. In parallel, methods like [5, 9–11, 98, 44] build on the idea that clusters should be formed in the feature spaces, and use clustering losses together with contrastive learning or transformation prediction tasks.

Most of the top-performing contrastive methods leverage data augmentations [12, 15, 28, 32, 51, 69]. As revealed by recent studies [4, 27, 70, 79], heavy data augmentations applied to the same image are crucial in learning useful representations, as they modulate the hardness of the self-supervised task via the *positive pair*. Our proposed hard negative mixing technique, on the other hand, is changing the hardness of the proxy task from the side of the *negatives*.

A few recent works discuss issues around the selection of negatives in contrastive self-supervised learning [8, 17, 36, 83, 85, 35]. Iscen *et al*. [36] mine hard negatives from a large set by focusing on the features that are neighbors with respect to the Euclidean distance, but not when using a manifold distance defined over the nearest neighbor graph. Interested in approximating the underlying "true" distribution of negative examples, Chuang *et al*. [17] present a *debiased* version of the contrastive loss, in an effort to mediate the effect of false negatives. Wu *et al*. [83] present a variational extension to the InfoNCE objective that is further coupled with modified strategies for negative sampling, *e.g*. restricting negative sampling to a region around the query. In concurrent works, Cao *et al*. [8] propose a weight update correction for negative samples to decrease GPU memory consumption caused by weight decay regularization, while in [35] the authors propose a new algorithm that generates more challenging positive and hard negative pairs, on-the-fly, by leveraging adversarial examples.

Mixing for contrastive learning. Mixup [93] and its numerous variants [62, 75, 77, 91] have been shown to be highly effective data augmentation strategies when paired with a cross-entropy loss for supervised and semi-supervised learning. Manifold mixup [75] is a feature-space regularizer that encourages networks to be less confident for interpolations of hidden states. The benefits of interpolating have only recently been explored for losses other than cross-entropy [39, 62, 97]. In [62], the authors propose using mixup in the image/pixel space for self-supervised learning; in contrast, we create query-specific synthetic points *on-the-fly* in the embedding space. This makes our method way more computationally efficient and able to show improved results at a smaller number of epochs. The Embedding Expansion [39] work explores interpolating between embeddings for supervised metric learning on fine-grained recognition tasks. The authors use uniform interpolation between two positive and negative points, create a set of synthetic points and then select the hardest pair as negative. In contrast, the proposed MoCHi has no need for class annotations, performs no selection for negatives and only samples a single random interpolation between multiple pairs. What is more, in this paper we go beyond mixing negatives and propose mixing the positive with negative features, to get even harder negatives, and achieve improved performance. Our work is also related to metric learning works that employ *generators* [20, 95]. Apart from not requiring labels, our method exploits the memory component and has no extra parameters or loss terms that need to be optimized.

3 Understanding hard negatives in unsupervised contrastive learning

3.1 Contrastive learning with memory

Let f be an encoder, *i.e*. a CNN for visual representation learning, that transforms an input image $\mathbf{x}$ to an *embedding* (or feature) vector $\mathbf{z} = f(\mathbf{x}), \mathbf{z} \in \mathbb{R}^d$. Further let Q be a "memory bank" of size K, *i.e*. a set of K embeddings in $\mathbb{R}^d$. Let the *query* $\mathbf{q}$ and *key* $\mathbf{k}$ embeddings form the positive pair, which is contrasted with every feature $\mathbf{n}$ in the bank of negatives (Q) also called the *queue* in [32]. A popular and highly successful loss function for contrastive learning [12, 32, 69] is the following:

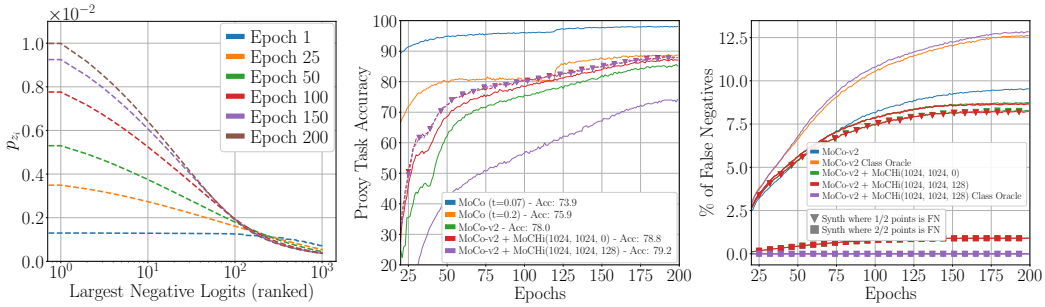
$$\mathcal{L}_{\mathbf{q},\mathbf{k},Q} = -\log \frac{\exp(\mathbf{q}^T\mathbf{k}/\tau)}{\exp(\mathbf{q}^T\mathbf{k}/\tau) + \sum_{\mathbf{n}\in Q}\exp(\mathbf{q}^T\mathbf{n}/\tau)}, \tag{1}$$

where τ is a temperature parameter and all embeddings are ℓ_2-normalized. In a number of recent successful approaches [12, 32, 51, 70] the query and key are the embeddings of two augmentations of the same image. The memory bank Q contains negatives for each positive pair, and may be defined as an "external" memory containing every other image in the dataset [51, 69, 84], a queue of the last batches [32], or simply be every other image in the current minibatch [12].

The log-likelihood function of Eq (1) is defined over the probability distribution created by applying a `softmax` function for each input/query $\mathbf{q}$. Let p_{z_i} be the matching probability for the query and feature $\mathbf{z}_i \in Z = Q \cup \{\mathbf{k}\}$, then the gradient of the loss with respect to the query $\mathbf{q}$ is given by:

$$\frac{\partial \mathcal{L}_{\mathbf{q},\mathbf{k},Q}}{\partial \mathbf{q}} = -\frac{1}{\tau}\left((1-p_k)\cdot\mathbf{k} - \sum_{\mathbf{n}\in Q} p_n \cdot \mathbf{n}\right), \quad \text{where } p_{z_i} = \frac{\exp(\mathbf{q}^T\mathbf{z}_i/\tau)}{\sum_{j\in Z}\exp(\mathbf{q}^T\mathbf{z}_j/\tau)}, \tag{2}$$

and p_k, p_n are the matching probability of the key and negative feature, *i.e*. for $\mathbf{z}_i = \mathbf{k}$ and for $\mathbf{z}_i = \mathbf{n}$, respectively. We see that the contributions of the positive and negative logits to the loss are identical to the ones for a $(K+1)$-way cross-entropy classification loss, where the logit for the key corresponds to the query's *latent class* [2] and all gradients are scaled by $1/\tau$.



(a) (b) (c)

Figure 2: Training on ImageNet-100 dataset. **(a)** A histogram of the 1024 highest matching probabilities $p_{z_i}, z_i \in Q$ for MoCo-v2 [15], across epochs; logits are ranked by decreasing order and each line shows the average value of the matching probability over all queries; **(b)** Accuracy on *the proxy task*, *i.e*. percentage of queries where we rank the key over all negatives. Lines with triangle markers for MoCHi variants correspond to the proxy task accuracy after discarding the synthetic hard negatives. **(c)** Percentage of false negatives (FN), *i.e*. negatives from the same class as the query, among the highest 1024 (negative) logits. when using a class oracle. Lines with triangle (resp. square) markers correspond to the percentage of synthetic points for which one (resp. both) mixed points are FN. For Purple and orange lines the class oracle was used during training to discard FN.

3.2 Hard negatives in contrastive learning

Hard negatives are critical for contrastive learning [2, 30, 36, 50, 63, 82, 89]. Sampling negatives from the same batch leads to a need for larger batches [12] while sampling negatives from a memory bank that contains every other image in the dataset requires the time consuming task of keeping a large memory up-to-date [51, 84]. In the latter case, a trade-off exists between the "freshness" of the memory bank representations and the computational overhead for re-computing them as the encoder keeps changing. The Momentum Contrast (or MoCo) approach of He et al. [32] offers a compromise between the two negative sampling extremes: it keeps a queue of the latest K features from the last batches, encoded with a second *key encoder* that trails the (main/query) encoder with a much higher momentum. For MoCo, the key feature $\mathbf{k}$ and all features in Q are encoded with the key encoder.

How hard are MoCo negatives? In MoCo [32] (resp. SimCLR [12]) the authors show that increasing the memory (resp. batch) size, is crucial to getting better and harder negatives. In Figure 2a we visualize how hard the negatives are during training for MoCo-v2, by plotting the highest 1024 matching probabilities p_i for ImageNet-100[1] and a queue of size $K = 16$k. We see that, although in the beginning of training (*i.e*. at epoch 0) the logits are relatively flat, as training progresses, fewer and fewer negatives offer significant contributions to the loss. This shows that most of the memory negatives are practically not helping a lot towards learning the proxy task.

On the difficulty of the proxy task. For MoCo [32], SimCLR [12], InfoMin [70], and other approaches that learn augmentation-invariant representations, we suspect the hardness of the proxy task to be directly correlated with the difficulty of the transformations set, *i.e*. hardness is modulated via the positive pair. We propose to experimentally verify this. In Figure 2b, we plot the *proxy task performance*, *i.e*. the percentage of queries where the key is ranked over all negatives, across training for MoCo [32] and MoCo-v2 [15]. MoCo-v2 enjoys a high performance gain over MoCo by three main changes: the addition of a Multilayer Perceptron (MLP) head, cosine learning rate schedule, and more challenging data augmentation. As we further discuss in the Appendix, only the latter of these three changes makes the proxy task harder to solve. Despite the drop in proxy task performance, however, further performance gains are observed for linear classification. In Section 4 we discuss how MoCHi gets a similar effect by modulating the proxy task through mixing harder negatives.

[1]In this section we study contrastive learning for MoCo [32] on ImageNet-100, a subset of ImageNet consisting of 100 classes introduced in [69]. See Section 5 for details on the dataset and experimental protocol.

3.3 A class oracle-based analysis

In this section, we analyze the negatives for contrastive learning using a class oracle, *i.e*. the ImageNet class label annotations. Let us define *false negatives* (FN) as all negative features in the memory Q, that correspond to images of the same class as the query. Here we want to first quantify false negatives from contrastive learning and then explore how they affect linear classification performance. What is more, by using class annotations, we can train a contrastive self-supervised learning *oracle*, where we measure performance at the downstream task (linear classification) after disregarding FN from the negatives of each query during training. This has connections to the recent work of [37], where a contrastive loss is used in a supervised way to form positive pairs from images sharing the same label. Unlike [37], our oracle uses labels only for discarding negatives with the same label for each query, *i.e*. without any other change to the MoCo protocol.

In Figure 2c, we quantify the percentage of false negatives for the oracle run and MoCo-v2, when looking at highest 1024 negative logits across training epochs. We see that a) in all cases, as representations get better, more and more FNs (same-class logits) are ranked among the top; b) by discarding them from the negatives queue, the class oracle version (purple line) is able to bring same-class embeddings closer. Performance results using the class oracle, as well as a supervised upper bound trained with cross-entropy are shown in the bottom section of Figure 1. We see that the MoCo-v2 oracle recovers part of the performance relative to the supervised case, *i.e*. 78.0 (MoCo-v2, 200 epochs) $\rightarrow$ 81.8 (MoCo-v2 oracle, 200 epochs) $\rightarrow$ 86.2 (supervised).

4 Feature space mixing of hard negatives

In this section we present an approach for synthesizing hard negatives, *i.e*. by *mixing* some of the hardest negative features of the contrastive loss or the hardest negatives with the query. We refer to the proposed hard negative mixing approach as **MoCHi**, and use the naming convention MoCHi (N, s, s'), that indicates the three important hyperparameters of our approach, to be defined below.

4.1 Mixing the hardest negatives

Given a query $\mathbf{q}$, its key $\mathbf{k}$ and negative/queue features $\mathbf{n} \in Q$ from a queue of size K, the loss for the query is composed of logits $l(\mathbf{z}_i) = \mathbf{q}^T \mathbf{z}_i / \tau$ fed into a softmax function. Let $\tilde{Q} = \{\mathbf{n}_1, \ldots, \mathbf{n}_K\}$ be the *ordered* set of all negative features, such that: $l(\mathbf{n}_i) > l(\mathbf{n}_j), \forall i < j$, *i.e*. the set of negative features sorted by decreasing similarity to that particular query feature.

For each query, we propose to synthesize s hard negative features, by creating convex linear combinations of pairs of its "hardest" existing negatives. We define the hardest negatives by truncating the ordered set $\tilde{Q}$, *i.e*. only keeping the first $N < K$ items. Formally, let $H = \{\mathbf{h}_1 \ldots, \mathbf{h}_s\}$ be the set of synthetic points to be generated. Then, a synthetic point $\mathbf{h}_k \in H$, would be given by:

$$\mathbf{h}_k = \frac{\tilde{\mathbf{h}}_k}{\|\tilde{\mathbf{h}}_k\|_2}, \text{ where } \tilde{\mathbf{h}}_k = \alpha_k \mathbf{n}_i + (1 - \alpha_k)\mathbf{n}_j, \tag{3}$$

$\mathbf{n}_i, \mathbf{n}_j \in \tilde{Q}^N$ are randomly chosen negative features from the set $\tilde{Q}^N = \{\mathbf{n}_1, \ldots, \mathbf{n}_N\}$ of the closest N negatives, $\alpha_k \in (0, 1)$ is a randomly chosen mixing coefficient and $\|\cdot\|_2$ is the ℓ_2-norm. After mixing, the logits $l(\mathbf{h}_k)$ are computed and appended as further *negative* logits for query $\mathbf{q}$. The process repeats for each query in the batch. Since all other logits $l(\mathbf{z}_i)$ are already computed, the extra computational cost only involves computing s dot products between the query and the synthesized features, which would be computationally equivalent to increasing the memory by $s << K$.

4.2 Mixing for even harder negatives

As we are creating hard negatives by convex combinations of the existing negative features, and if one disregards the effects of the ℓ_2-normalization for the sake of this analysis, the generated features will lie inside the convex hull of the hardest negatives. Early during training, where in most cases there is no linear separability of the query with the negatives, this synthesis may result in negatives much harder than the current. As training progresses, and assuming that linear separability is achieved, synthesizing features this way does not necessarily create negatives harder than the hardest ones present; it does however still *stretch* the space around the query, pushing the memory negatives further

and increasing the uniformity of the space (see Section 4.3). This space stretching effect around queries is also visible in the t-SNE projection of Figure 1.

To explore our intuition to the fullest, we further propose to mix the *query* with the hardest negatives to get even harder negatives for the proxy task. We therefore further synthesize s' synthetic hard negative features for each query, by mixing its feature with a randomly chosen feature from the hardest negatives in set $\tilde{Q}^N$. Let $H' = \{\mathbf{h}'_1 \dots, \mathbf{h}'_{s'}\}$ be the set of synthetic points to be generated by mixing the query and negatives, then, similar to Eq. (3), the synthetic points $\mathbf{h}'_k = \tilde{\mathbf{h}}'_k / \|\tilde{\mathbf{h}}'_k\|_2$, where $\tilde{\mathbf{h}}'_k = \beta_k \mathbf{q} + (1 - \beta_k)\mathbf{n}_j$, and $\mathbf{n}_j$ is a randomly chosen negative feature from $\tilde{Q}^N$, while $\beta_k \in (0, 0.5)$ is a randomly chosen mixing coefficient for the query. Note that $\beta_k < 0.5$ guarantees that the query's contribution is always smaller than the one of the negative. Same as for the synthetic features created in Section 4.1, the logits $l(\mathbf{h}'_k)$ are computed and added as further *negative* logits for query $\mathbf{q}$. Again, the extra computational cost only involves computing s' dot products between the query and negatives. In total, the computational overhead of MoCHi is essentially equivalent to increasing the size of the queue/memory by $s + s' << K$.

4.3 Discussion and analysis of MoCHi

Recent approaches like [12, 15] use a Multi-layer Perceptron (MLP) head instead of a linear layer for the embeddings that participate in the contrastive loss. This means that the embeddings whose dot products contribute to the loss, are not the ones used for target tasks–a lower-layer embedding is used instead. Unless otherwise stated, we follow [12, 15] and use a 2-layer MLP head on top of the features we use for downstream tasks. We always mix hard negatives in the space of the loss.

Is the proxy task more difficult? Figure 2b shows the *proxy task* performance for two variants of MoCHi, when the synthetic features are included (lines with no marker) and without (lines with triangle marker). We see that when mixing only pairs of negatives ($s' = 0$, green lines), the model does learn faster, but in the end the proxy task performance is similar to the baseline case. In fact, as features converge, we see that $\max l(\mathbf{h}_k) < \max l(\mathbf{n}_j), \mathbf{h}_k \in H, \mathbf{n}_j \in \tilde{Q}^N$. This is however not the case when synthesizing negatives by further mixing them *with* the query. As we see from Figure 2b, at the end of training, $\max l(\mathbf{h}'_k) > \max l(\mathbf{n}_j), \mathbf{h}'_k \in H'$, *i.e.* although the final performance for the proxy task when discarding the synthetic negatives is similar to the MoCo-v2 baseline (red line with triangle marker), when they are taken into account, the final performance is much lower (red line without markers). Through MoCHi we are able to modulate the hardness of the proxy task through the hardness of the negatives; in the next section we experimentally ablate that relationship.

Oracle insights for MoCHi. From Figure 2c we see that the percentage of synthesized features obtained by mixing two false negatives (lines with square markers) increases over time, but remains very small, *i.e.* around only 1%. At the same time, we see that about 8% of the synthetic features are fractionally false negatives (lines with triangle markers), *i.e.* at least one of its two components is a false negative. For the oracle variants of MoCHi, we also do not allow false negatives to participate in synthesizing hard negatives. From Table 1 we see that not only the MoCHi oracle is able to get a higher upper bound (82.5 vs 81.8 for MoCo-v2), further closing the difference to the cross entropy upper bound, but we also show in the Appendix that, after longer training, the MoCHi oracle is able to recover *most* of the performance loss versus using cross-entropy, *i.e.* 79.0 (MoCHi, 200 epochs) $\rightarrow$ 82.5 (MoCHi oracle, 200 epochs) $\rightarrow$ 85.2 (MoCHi oracle, 800 epochs) $\rightarrow$ 86.2 (supervised).

It is noteworthy that by the end of training, MoCHi exhibits slightly lower percentage of false negatives in the top logits compared to MoCo-v2 (rightmost values of Figure 2c). This is an interesting result: MoCHi adds synthetic negative points that are (at least partially) false negatives and is pushing embeddings of the same class apart, but at the same time it exhibits higher performance for linear classification on ImageNet-100. That is, it seems that although the absolute similarities of same-class features may decrease, the method results in a more linearly separable space. This inspired us to further look into how having synthetic hard negatives impacts the *utilization* of the embedding space.

Measuring the utilization of the embedding space. Very recently, Wang and Isola [79] presented two losses/metrics for assessing contrastive learning representations. The first measures the *alignment* of the representation on the hypersphere, *i.e.* the absolute distance between representations with the same label. The second measures the *uniformity* of their distribution on the hypersphere, through

measuring the logarithm of the average pairwise Gaussian potential between all embeddings. In Figure 3c, we plot these two values for a number of models, when using features from all images in the ImageNet-100 validation set. We see that MoCHi highly improves the uniformity of the representations compared to both MoCo-v2 and the supervised models. This further supports our hypothesis that MoCHi allows the proxy task to learn to better utilize the embedding space. In fact, we see that the supervised model leads to high alignment but very low uniformity, denoting features targeting the classification task. On the other hand, MoCo-v2 and MoCHi have much better spreading of the underlying embedding space, which we experimentally know leads to more *generalizable* representations, *i.e*. both MoCo-v2 and MoCHi outperform the supervised ImageNet-pretrained backbone for transfer learning (see Figure 3c).

5 Experiments

We learn representations on two datasets, the common ImageNet-1K [61], and its smaller ImageNet-100 subset, also used in [62, 69]. All runs of MoCHi are based on MoCo-v2. We developed our approach on top of the official public implementation of MoCo-v2[2] and reproduced it on our setup; other results are copied from the respective papers. We run all experiments on 4 GPU servers. For linear classification on ImageNet-100 (resp. ImageNet-1K), we follow the common protocol and report results on the validation set. We report performance after learning linear classifiers for 60 (resp. 100) epochs, with an initial learning rate of 10.0 (30.0), a batch size of 128 (resp. 512) and a step learning rate schedule that drops at epochs 30, 40 and 50 (resp. 60, 80). For training we use $K = 16$k (resp. $K = 65$k). For MoCHi, we also have a warm-up of 10 (resp. 15) epochs, *i.e*. for the first epochs we do not synthesize hard negatives. For ImageNet-1K, we report accuracy for a single-crop testing. For object detection on PASCAL VOC [21] we follow [32] and fine-tune a Faster R-CNN [60], R50-C4 on `trainval07+12` and test on `test2007`. We use the open-source `detectron2`[3] code and report the common AP, AP50 and AP75 metrics. Similar to [32], we do *not* perform hyperparameter tuning for the object detection task. See the Appendix for more implementation details.

Method	Top1 % ($\pm\sigma$)	diff (%)
MoCo [32]	73.4	
MoCo + iMix [62]	74.2‡	↑0.8
CMC [69]	75.7	
CMC + iMix [62]	75.9‡	↑0.2
MoCo [32]* ($t = 0.07$)	74.0	
MoCo [32]* ($t = 0.2$)	75.9	
MoCo-v2 [15]*	78.0 ($\pm$0.2)	
+ MoCHi (1024, 1024, 128)	**79.0** ($\pm$0.4)	↑**1.0**
+ MoCHi (1024, 256, 512)	**79.0** ($\pm$0.4)	↑**1.0**
+ MoCHi (1024, 128, 256)	**78.9** ($\pm$0.5)	↑**0.9**
Using Class Oracle		
MoCo-v2*	81.8	
+ MoCHi (1024, 1024, 128)	82.5	
Supervised (Cross Entropy)	*86.2*	

Table 1: Results on ImageNet-100 after training for 200 epochs. The bottom section reports results when using a class oracle (see Section 3.3). * denotes reproduced results, ‡ denotes results visually extracted from Figure 4 in [62]. The parameters of MoCHi are (N, s, s').

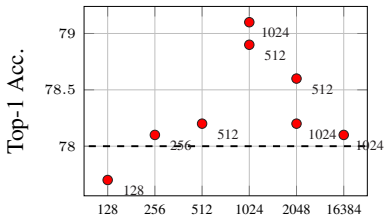
A note on reporting variance in results. It is unfortunate that many recent self-supervised learning papers do not discuss variance ; in fact only papers from highly resourceful labs [32, 12, 70] report averaged results, but not always the variance. This is generally understandable, as *e.g*. training and evaluating a ResNet-50 model on ImageNet-1K using 4 V100 GPUs take about 6-7 days. In this paper, we tried to verify the variance of our approach for a) self-supervised pre-training on ImageNet-100, *i.e*. we measure the variance of MoCHi runs by training a model multiple times from scratch (Table 1), and b) the variance in the fine-tuning stage for PASCAL VOC and COCO (Tables 2, 3). It was unfortunately computationally infeasible for us to run multiple MoCHi pre-training runs for ImageNet-1K. In cases where standard deviation is presented, it is measured over at least 3 runs.

5.1 Ablations and results

We performed extensive ablations for the most important hyperparameters of MoCHi on ImageNet-100 and some are presented in Figures 3a and 3b, while more can be found in the Appendix. In general we see that multiple MoCHi configurations gave consistent gains over the MoCo-v2 baseline [15] for linear classification, with the top gains presented in Figure 1 (also averaged over 3 runs). We further show performance for different values of N and s in Figure 3a and a table of gains for $N = 1024$ in

[2] https://github.com/facebookresearch/moco

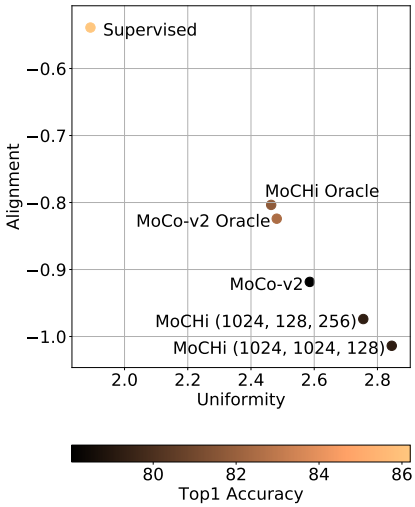
[3] https://github.com/facebookresearch/detectron2



(a) Accuracy when varying N (x-axis) and s.

s \ s'	0	128	256	512
0	0.0	+0.7	+0.9	+1.0
128	+0.8	+0.4	+1.1	+0.3
256	+0.3	+0.7	+0.3	+1.0
512	+0.9	+0.8	+0.6	+0.4
1024	+0.8	+1.0	+0.7	+0.6

(b) Accuracy gains over MoCo-v2 when $N = 1024$.



(c) Alignment and uniformity.

Figure 3: Results on the validation set of ImageNet-100. **(a)** Top-1 Accuracy for different values of N (x-axis) and s; the dashed black line is MoCo-v2. **(b)** Top-1 Accuracy gains (%) over MoCo-v2 (top-left cell) when $N = 1024$ and varying s (rows) and s' (columns). **(c)** Alignment and uniformity metrics [79]. The x/y axes correspond to $-\mathcal{L}_{uniform}$ and $-\mathcal{L}_{align}$, respectively.

Figure 3b; we see that a large number of MoCHi combinations give consistent performance gains. Note that the results in these two tables are *not averaged* over multiple runs (for MoCHi combinations where we had multiple runs, only the first run is presented for fairness). In other ablations (see Appendix), we see that MoCHi achieves gains (+0.7%) over MoCo-v2 also when training for 100 epochs. Table 1 presents comparisons between the best-performing MoCHi variants and reports gains over the MoCo-v2 baseline. We also compare against the published results from [62] a recent method that uses mixup in pixel space to synthesize harder images.

Comparison with the state of the art on ImageNet-1K, PASCAL VOC and COCO. In Table 2 we present results obtained after training on the ImageNet-1K dataset. Looking at the average negative logits plot and because both the queue and the dataset are about an order of magnitude larger for this training dataset we mostly experiment with smaller values for N than in ImageNet-100. Our main observations are the following: a) MoCHi does not show performance gains over MoCo-v2 for linear classification on ImageNet-1K. We attribute this to the biases induced by training with hard negatives on the same dataset as the downstream task: Figures 3c and 2c show how hard negative mixing reduces alignment and increases uniformity for the dataset that is used during training. MoCHi still retains state-of-the-art performance. b) MoCHi helps the model learn faster and achieves high performance gains over MoCo-v2 for transfer learning after only 100 epochs of training. c) The harder negative strategy presented in Section 4.2 helps a lot for shorter training. d) In 200 epochs MoCHi can achieve *performance similar to MoCo-v2 after 800 epochs* on PASCAL VOC. e) From all the MoCHi runs reported in Table 2 as well as in the Appendix, we see that performance gains are consistent across multiple hyperparameter configurations.

In Table 3 we present results for object detection and semantic segmentation on the COCO dataset [45]. Following He et al. [32], we use Mask R-CNN [31] with a C4 backbone, with batch normalization tuned and synchronize across GPUs. The image scale is in [640, 800] pixels during training and is 800 at inference. We fine-tune all layers end-to-end on the `train2017` set (118k images) and evaluate on `val2017`. We adopt feature normalization as in [32] when fine-tuning. MoCHi and MoCo use the same hyper-parameters as the ImageNet supervised counterpart (*i.e.* we did not do any method-specific tuning). From Table 3 we see that MoCHi displays consistent gains over both the supervised baseline and MoCo-v2, for both 100 and 200 epoch pre-training. In fact, MoCHi is able to reach the AP performance similar to supervised pre-training for instance segmentation (33.2) after only 100 epochs of pre-training.

Method	IN-1k Top1	VOC 2007 AP_{50}	VOC 2007 AP	VOC 2007 AP_{75}
100 epoch training				
MoCo-v2 [15]*	63.6	80.8 (±0.2)	53.7 (±0.2)	59.1 (±0.3)
+ MoCHi (256, 512, 0)	63.9	81.1 (±0.1) (↑0.4)	54.3 (±0.3) (↑0.7)	60.2 (±0.1) (↑1.2)
+ MoCHi (256, 512, 256)	63.7	**81.3** (±0.1) (↑**0.6**)	54.6 (±0.3) (↑1.0)	60.7 (±0.8) (↑1.7)
+ MoCHi (128, 1024, 512)	63.4	81.1 (±0.1) (↑0.4)	**54.7** (±0.3) (↑**1.1**)	**60.9** (±0.1) (↑**1.9**)
200 epoch training				
SimCLR [12] (8k batch size, from [15])	66.6			
MoCo + Image Mixture [62]	60.8	76.4		
InstDis [84]†	59.5	80.9	55.2	61.2
MoCo [32]	60.6	81.5	55.9	62.6
PIRL [51]†	61.7	81.0	55.5	61.3
MoCo-v2 [15]	67.7	82.4	57.0	63.6
InfoMin Aug. [70]	70.1	82.7	**57.6**	**64.6**
MoCo-v2 [15]*	67.9	82.5 (±0.2)	56.8 (±0.1)	63.3 (±0.4)
+ MoCHi (1024, 512, 256)	68.0	82.3 (±0.2) (↓0.2)	56.7 (±0.2) (↓0.1)	63.8 (±0.2) (↑0.5)
+ MoCHi (512, 1024, 512)	67.6	82.7 (±0.1) (↑0.2)	57.1 (±0.1) (↑0.3)	64.1 (±0.3) (↑0.8)
+ MoCHi (256, 512, 0)	67.7	**82.8** (±0.2) (↑0.3)	57.3 (±0.2) (↑0.5)	64.1 (±0.1) (↑0.8)
+ MoCHi (256, 512, 256)	67.6	82.6 (±0.2) (↑0.1)	57.2 (±0.3) (↑0.4)	64.2 (±0.5) (↑0.9)
+ MoCHi (256, 2048, 2048)	*67.0*	82.5 (±0.1) (0.0)	57.1 (±0.2) (↑0.3)	64.4 (±0.2) (↑1.1)
+ MoCHi (128, 1024, 512)	*66.9*	82.7 (±0.2) (↑0.2)	57.5 (±0.3) (↑0.7)	64.4 (±0.4) (↑1.1)
800 epoch training				
SvAV [11]	*75.3*	82.6	56.1	62.7
MoCo-v2 [15]	71.1	82.5	**57.4**	64.0
MoCo-v2[15]*	69.0	82.7 (±0.1)	56.8 (±0.2)	63.9 (±0.7)
+ MoCHi (128, 1024, 512)	68.7	**83.3** (±0.1) (↑0.6)	57.3 (±0.2) (↑0.5)	**64.2** (±0.4) (↑0.3)
1000 epoch training				
MoCo-v2[15] + MoCHi (512, 1024, 512)	70.6	83.2 (±0.3)	58.4 (±0.2)	65.5 (±0.6)
MoCo-v2[15] + MoCHi (128, 1024, 512)	69.8	**83.4** (±0.2)	**58.7** (±0.1)	**65.9** (±0.2)
Supervised [32]	76.1	81.3	53.5	58.8

Table 2: Results for linear classification on **ImageNet-1K** and object detection on **PASCAL VOC** with a ResNet-50 backbone. Wherever standard deviation is reported, it refers to multiple runs for the fine-tuning part. For MoCHi runs we also report in parenthesis the difference to MoCo-v2. * denotes reproduced results. † results are copied from [32]. We **bold** (resp. underline) the highest results overall (resp. for MoCHi).

Pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
Supervised [32]	38.2	58.2	41.6	33.3	54.7	35.2
	100 epoch pre-training					
MoCo-v2 [15]*	37.0 (±0.1)	56.5 (±0.3)	39.8 (±0.1)	32.7 (±0.1)	53.3 (±0.2)	34.3 (±0.1)
+ MoCHi (256, 512, 0)	37.5 (±0.1) (↑0.5)	57.0 (±0.1) (↑0.5)	40.5 (±0.2) (↑0.7)	33.0 (±0.1) (↑0.3)	53.9 (±0.2) (↑0.6)	34.9 (±0.1) (↑0.6)
+ MoCHi (128, 1024, 512)	**37.8** (±0.1) (↑**0.8**)	**57.2** (±0.0) (↑**0.7**)	**40.8** (±0.2) (↑**1.0**)	**33.2** (±0.0) (↑**0.5**)	54.0 (±0.2) (↑**0.7**)	35.4 (±0.1) (↑**1.1**)
	200 epoch pre-training					
MoCo [32]	38.5	58.3	41.6	33.6	54.8	35.6
MoCo (1B image train) [32]	39.1	58.7	42.2	34.1	55.4	36.4
InfoMin Aug. [70]	39.0	58.5	42.0	34.1	55.2	36.3
MoCo-v2 [15]*	39.0 (±0.1)	58.6 (±0.1)	41.9(±0.3)	34.2 (±0.1)	55.4 (±0.1)	36.2 (±0.2)
+ MoCHi (256, 512, 0)	39.2 (±0.1) (↑0.2)	58.8 (±0.1) (↑0.2)	42.4 (±0.2) (↑0.5)	34.4 (±0.1) (↑0.2)	55.6 (±0.1) (↑0.2)	36.7 (±0.1) (↑0.5)
+ MoCHi (128, 1024, 512)	39.2 (±0.1) (↑0.2)	58.9 (±0.2) (↑0.3)	42.4 (±0.3) (↑0.5)	34.3 (±0.1) (↑0.2)	55.5 (±0.1) (↑0.1)	36.6 (±0.1) (↑0.4)
+ MoCHi (512, 1024, 512)	**39.4** (±0.1) (↑**0.4**)	**59.0** (±0.1) (↑**0.4**)	**42.7** (±0.1) (↑**0.8**)	**34.5** (±0.0) (↑**0.3**)	**55.7** (±0.2) (↑**0.3**)	**36.7** (±0.1) (↑**0.5**)

Table 3: Object detection and instance segmentation results on **COCO** with the $\times 1$ training schedule and a C4 backbone. * denotes reproduced results.

6 Conclusions

In this paper we analyze a state-of-the-art method for self-supervised contrastive learning and identify the need for harder negatives. Based on that observation, we present a hard negative mixing approach that is able to improve the quality of representations learned in an unsupervised way, offering better transfer learning performance as well as a better utilization of the embedding space. What is more, we show that we are able to learn generalizable representations *faster*, something important considering the high compute cost of self-supervised learning. Although the hyperparameters needed to get maximum gains are specific to the training set, we find that multiple MoCHi configurations provide considerable gains, and that hard negative mixing consistently has a positive effect on transfer

learning performance. Pretrained models are publicly available to download from our project page at https://europe.naverlabs.com/mochi.

Broader Impact

Self-supervised tasks and dataset bias. Prominent voices in the field advocate that self-supervised learning will play a central role during the next years in the field of AI. Not only representations learned using self-supervised objectives directly reflect the biases of the underlying dataset, but also it is the responsibility of the scientist to explicitly try to minimize such biases. Given that, the larger the datasets, the harder it is to properly investigate biases in the corpus, we believe that notions of *fairness* need to be explicitly tackled during the self-supervised *optimization*, *e.g.* by regulating fairness on protected attributes. This is especially important for systems whose decisions affect humans and/or their behaviours.

Self-supervised learning, compute and impact on the environment. On the one hand, self-supervised learning involves training large models on very large datasets, on long periods of time. As we also argue in the main paper, the computational cost of every self-supervised learning paper is very high: pre-training for 200 epochs on the relatively small ImageNet-1K dataset requires around 24 GPU days (6 days on 4 GPUs). In this paper we show that, by mining harder negatives, one can get higher performance after training for fewer epochs; we believe that it is indeed important to look deeper into self-supervised learning approaches that utilize the training dataset better and learn generalizable representations faster.

Looking at the bigger picture, however, we believe that research in self-supervised learning is highly justified in the long run, despite its high computational cost, for two main reasons. First, the goal of self-supervised learning is to produce models whose representations generalize better and are therefore potentially useful for many subsequent tasks. Hence, having strong models pre-trained by self-supervision would reduce the environmental impact of deploying to multiple new downstream tasks. Second, representations learned from huge corpora have been shown to improve results when directly fine-tuned, or even used as simple feature extractors, on smaller datasets. Most socially minded applications and tasks fall in this situation where they have to deal with limited annotated sets, because of a lack of funding, hence they would directly benefit from making such pretraining models available. Given the considerable budget required for large, high quality datasets, we foresee that strong generalizable representations will greatly benefit socially mindful tasks more than *e.g.* a multi-billion dollar industry application, where the funding to get large clean datasets already exists.

Acknowledgements

This work is part of MIAI@Grenoble Alpes (ANR-19-P3IA-0003).

References

[1] H. Alwassel, D. Mahajan, L. Torresani, B. Ghanem, and D. Tran. Self-supervised learning by cross-modal audio-video clustering. *NeurIPS*, 2020. 18

[2] S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, and N. Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *ICML*, 2019. 3, 4

[3] Y. M. Asano, M. Patrick, C. Rupprecht, and A. Vedaldi. Labelling unlabelled videos from scratch with multi-modal self-supervision. In *NeurIPS*, 2020. 18

[4] Y. M. Asano, C. Rupprecht, and A. Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. *ICLR*, 2020. 2

[5] Y. M. Asano, C. Rupprecht, and A. Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 2, 18, 21

[6] P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019. 18

[7] Q. Cai, Y. Wang, Y. Pan, T. Yao, and T. Mei. Joint contrastive learning with infinite possibilities. In *NeurIPS*, 2020. 19

[8] Y. Cao, Z. Xie, B. Liu, Y. Lin, Z. Zhang, and H. Hu. Parametric instance classification for unsupervised visual feature learning. In *NeurIPS*, 2020. 3

[9] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 2, 18

[10] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019. 17, 21

[11] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020. 2, 9, 18, 19, 21

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 2, 3, 4, 6, 7, 9, 18, 19, 21

[13] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 2

[14] X. Chen and K. He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020. 19

[15] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2, 4, 6, 7, 9, 15, 16, 19, 20, 21

[16] Y. Cheng, R. Wang, Z. Pan, R. Feng, and Y. Zhang. Look, listen, and attend: Co-attention network for self-supervised audio-visual representation learning. *ACM Multimedia*, 2020. 18

[17] C.-Y. Chuang, J. Robinson, L. Yen-Chen, A. Torralba, and S. Jegelka. Debiased contrastive learning. In *NeurIPS*, 2020. 3

[18] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 2

[19] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NeurIPS*, 2014. 2

[20] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou. Deep adversarial metric learning. In *CVPR*, 2018. 3, 19

[21] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 7

[22] W. Falcon and K. Cho. A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104*, 2020. 18

[23] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017. 18

[24] W. V. Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. V. Gool. Scan: Learning to classify images without labels. *ECCV*, 2020. 18

[25] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2

[26] L. Gomez, Y. Patel, M. Rusiñol, D. Karatzas, and C. Jawahar. Self-supervised learning of visual features through embedding images into text topic spaces. In *CVPR*, 2017. 18

[27] R. Gontijo-Lopes, S. J. Smullin, E. D. Cubuk, and E. Dyer. Affinity and diversity: Quantifying mechanisms of data augmentation. *arXiv preprint arXiv:2002.08973*, 2020. 1, 2

[28] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *NeurIPS*, 2020. 2, 19

[29] T. Han, W. Xie, and A. Zisserman. Memory-augmented dense predictive coding for video representation learning. *ECCV*, 2020. 18

[30] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond, et al. Smart mining for deep metric learning. In *ICCV*, 2017. 4

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN]. In *ICCV*, 2017. 8

[32] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2, 3, 4, 7, 8, 9, 15, 18, 19, 20, 21

[33] R. D. Hjelm and P. Bachman. Representation learning with video deep infomax. *arXiv preprint arXiv:2007.13278*, 2020. 18

[34] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019. 18

[35] C.-H. Ho and N. Vasconcelos. Contrastive learning with adversarial examples. In *NeurIPS*, 2020. 3

[36] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Mining on manifolds: Metric learning without labels. In *CVPR*, 2018. 3, 4

[37] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020. 5, 19

[38] M. Kim, J. Tack, and S. J. Hwang. Adversarial self-supervised contrastive learning. *NeurIPS*, 2020. 18

[39] B. Ko and G. Gu. Embedding expansion: Augmentation in embedding space for deep metric learning. *CVPR*, 2020. 3, 19

[40] A. Kolesnikov, X. Zhai, and L. Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, 2019. 2

[41] B. Korbar, D. Tran, and L. Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *NeurIPS*, 2018. 18

[42] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017. 18

[43] J. D. Lee, Q. Lei, N. Saunshi, and J. Zhuo. Predicting what you already know helps: Provable self-supervised learning. *arXiv preprint arXiv:2008.01064*, 2020. 18

[44] J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020. 2, 17, 18, 21

[45] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 8

[46] H. Liu and P. Abbeel. Hybrid discriminative-generative training via contrastive learning. *arXiv preprint arXiv:2007.09070*, 2020. 19

[47] S. Löwe, P. O'Connor, and B. Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *NeurIPS*, 2019. 18

[48] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 2008. 2

[49] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman. End-to-End Learning of Visual Representations from Uncurated Instructional Videos. In *CVPR*, 2020. 18

[50] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *NeurIPS*, 2017. 4

[51] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. *CVPR*, 2020. 2, 3, 4, 9, 18, 21

[52] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016. 18

[53] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 2

[54] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

[55] A. E. Orhan, V. V. Gupta, and B. M. Lake. Self-supervised learning through the eyes of a child. In *NeurIPS*, 2020. 18

[56] M. Patacchiola and A. Storkey. Self-supervised relational reasoning for representation learning. In *NeurIPS*, 2020. 18

[57] M. Patrick, Y. M. Asano, R. Fong, J. F. Henriques, G. Zweig, and A. Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv preprint arXiv:2003.04298*, 2020. 18

[58] S. Purushwalkam and A. Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *NeurIPS*, 2020. 18

[59] S.-A. Rebuffi, S. Ehrhardt, K. Han, A. Vedaldi, and A. Zisserman. Lsd-c: Linearly separable deep clusters. *arXiv preprint arXiv:2006.10039*, 2020. 18

[60] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 7, 17

[61] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 7

[62] Z. Shen, Z. Liu, Z. Liu, M. Savvides, and T. Darrell. Rethinking image mixture for unsupervised visual representation learning. *arXiv preprint arXiv:2003.05438*, 2020. 3, 7, 8, 9, 17, 20, 21

[63] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 4

[64] J. Song and S. Ermon. Multi-label contrastive predictive coding. *NeurIPS*, 2020. 18

[65] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015. 18

[66] C. Sun, F. Baradel, K. Murphy, and C. Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv:1906.05743*, 2019. 18

[67] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. VideoBERT: A joint model for video and language representation learning. In *ICCV*, 2019. 18

[68] L. Tao, X. Wang, and T. Yamasaki. Self-supervised video representation learning using inter-intra contrastive framework. *ACM Multimedia*, 2020. 18

[69] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2, 3, 4, 7, 17, 20

[70] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning. In *NeurIPS*, 2020. 1, 2, 3, 4, 7, 9, 19, 21

[71] Y. Tian, L. Yu, X. Chen, and S. Ganguli. Understanding self-supervised learning with dual deep networks. *arXiv preprint arXiv:2010.00578*, 2020. 18, 19

[72] P. Tokmakov, M. Hebert, and C. Schmid. Unsupervised learning of video representations via dense trajectory clustering. *arXiv preprint arXiv:2006.15731*, 2020. 18

[73] Y.-H. H. Tsai, Y. Wu, R. Salakhutdinov, and L.-P. Morency. Demystifying self-supervised learning: An information-theoretical framework. *arXiv preprint arXiv:2006.05576*, 2020. 18

[74] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic. On mutual information maximization for representation learning. *ICLR*, 2020. 18

[75] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019. 2, 3

[76] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz. Interpolation consistency training for semi-supervised learning. *IJCAI*, 2019. 2

[77] D. Walawalkar, Z. Shen, Z. Liu, and M. Savvides. Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification. In *ICASSP*, 2020. 3

[78] J. Wang, J. Jiao, and Y.-H. Liu. Self-supervised video representation learning by pace prediction. *arXiv preprint arXiv:2008.05861*, 2020. 18

[79] T. Wang and P. Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020. 2, 6, 8, 14, 18

[80] C. Wei, H. Wang, W. Shen, and A. Yuille. Co2: Consistent contrast for unsupervised visual representation learning. *arXiv preprint arXiv:2010.02217*, 2020. 18, 21

[81] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *CVPR*, 2018. 18

[82] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl. Sampling matters in deep embedding learning. In *ICCV*, 2017. 4

[83] M. Wu, C. Zhuang, M. Mosse, D. Yamins, and N. Goodman. On mutual information in contrastive learning for visual representations. *arXiv preprint arXiv:2005.13149*, 2020. 3, 18

[84] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 2, 3, 4, 9, 21

[85] J. Xie, X. Zhan, Z. Liu, Y. S. Ong, and C. C. Loy. Delving into inter-image invariance for unsupervised visual representations. *arXiv preprint arXiv:2008.11702*, 2020. 3, 21

[86] Y. Xiong, M. Ren, and R. Urtasun. Loco: Local contrastive representation learning. *NeurIPS*, 2020. 18

[87] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*, 2019. 18

[88] H. Xu, H. Xiong, and G.-J. Qi. K-shot contrastive learning of visual features with multiple instance augmentations. *arXiv preprint arXiv:2007.13310*, 2020. 19

[89] H. Xuan, A. Stylianou, X. Liu, and R. Pless. Hard negative examples are hard, but useful. In *ECCV*, 2020. 4

[90] X. Yan, I. Misra, A. Gupta, D. Ghadiyaram, and D. Mahajan. Clusterfit: Improving generalization of visual representations. In *CVPR*, 2020. 18

[91] 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 *ICCV*, 2019. 2, 3

[92] X. Zhan, J. Xie, Z. Liu, Y.-S. Ong, and C. C. Loy. Online deep clustering for unsupervised representation learning. In *CVPR*, 2020. 18

[93] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 2, 3

[94] N. Zhao, Z. Wu, R. W. Lau, and S. Lin. What makes instance discrimination good for transfer learning? *arXiv preprint arXiv:2006.06606*, 2020. 19

[95] W. Zheng, Z. Chen, J. Lu, and J. Zhou. Hardness-aware deep metric learning. In *CVPR*, 2019. 3, 19

[96] H. Zhong, C. Chen, Z. Jin, and X.-S. Hua. Deep robust clustering by contrastive learning. *arXiv preprint arXiv:2008.03030*, 2020. 18

[97] H.-Y. Zhou, S. Yu, C. Bian, Y. Hu, K. Ma, and Y. Zheng. Comparing to learn: Surpassing imagenet pretraining on radiographs by comparing image representations. In *MICCAI*, 2020. 3

[98] C. Zhuang, A. L. Zhai, and D. Yamins. Local aggregation for unsupervised learning of visual embeddings. In *ICCV*, 2019. 2, 18

Appendices

Appendix A Details for the uniformity experiment

The uniformity experiment is based on Wang and Isola [79]. We follow the same definitions of the losses/metrics as presented in the paper. The alignment loss is given by:

$$\mathcal{L}_{\mathsf{align}}(f;\alpha) := - \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim p_{\mathsf{pos}}} \left[\|f(\mathbf{x}) - f(\mathbf{y})\|_2^\alpha\right], \quad \alpha > 0,$$

while the uniformity loss is given by:

$$\mathcal{L}_{\mathsf{uniform}}(f;t) := \log \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y} \overset{\text{i.i.d.}}{\sim} p_{\mathsf{data}}} \left[e^{-t\|f(\mathbf{x})-f(\mathbf{y})\|_2^2}\right], \quad t > 0,$$

where α, t are weighting parameters and f is the feature encoder (*i.e.* minus the MLP head for MoCo-v2 and MoCHi). We set $\alpha = 2$ and $t = 2$. All features were L2-normalized, as the metrics are defined on the hypersphere. p_{pos} denotes the joint distribution of pairs of positive samples, and p_{data} is the distribution of the data. Note that p_{pos} is task-specific; here we use the class oracle, *i.e.* the ImageNet-100 labels, to define the positive samples. We use the publicly available implementation supplied by the authors[4]; we modify the alignment implementation to reflect the fact that we obtain the positives based on the class oracle. In the Figure, we report the two metrics ($-\mathcal{L}_{uniform}$ and $-\mathcal{L}_{align}$) for models trained on ImageNet-100 using all embeddings of the validation set.

Appendix B Further analysis on hard negative mixing

B.1 Proxy task: Effect of MLP and Stronger Augmentation

Following our discussion in Section 3, we wanted to verify that hardness of the proxy task for MoCo [32] is directly correlated to the difficulty of the transformations set, *i.e.* proxy task hardness can modulated via the positive pair. In Figure 4, we plot the *proxy task performance*, *i.e.* the percentage of queries where the key is ranked over all negatives, across training for MoCo [32], MoCo-v2 [15] and some variants inbetween. In Figure 4, we track the proxy task performance when progressively moving from MoCo to MoCo-v2, *i.e.* a) switching to a cosine learning rate schedule (gray line–no noticeable change in performance after 200 epochs); b) adding a Multilayer Perceptron head (cyan line–no noticeable change in performance after 200 epochs); c) adding a more challenging data augmentation (green line–a drop in proxy task performance). The latter is equivalent to MoCo-v2. For completeness we further show a MoCHi run with a large number of synthetic features (red line–large drop in in proxy task performance).

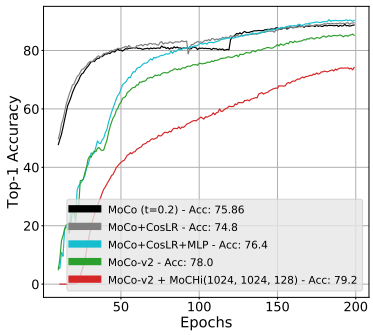


Figure 4: Proxy task performance over 200 epochs of training on ImageNet-100. For all methods we use the same $\tau = 0.2$.

It is worth noting that the temperature τ of the softmax is a hyper-parameter that highly affects the rate of learning and therefore performance in the proxy task. As mentioned above, all results in Figure 4 are for the same $\tau = 0.2$.

B.2 Hard negative mixing variants not discussed in the main text

While developing MoCHi, we considered a number of different mixing strategies in feature space. Many of those resulted in lower performance while others performed on par but were unnecessarily more complicated. We found the two strategies presented in Sections 4.1 and 4.2 of the main paper to be both the best performing and also complementary. Here, we briefly mention some other ideas that didn't make the cut.

Mixing using keys instead of queries. For MoCHi, the "top" negatives are defined via the negative logits, *i.e.* how far each memory negative is to a *query*. We also experimented when the ranking comes from a *key*, *i.e.* using the key to define the ordering of the set $\tilde{Q}$. We ablated this for both when mixing pairs of negative as well as when mixing the query with negatives. In the vast majority of the cases, results were on average about 0.2% lower, across multiple configurations. Note that this would also involve having to compute the dot products of the key with the memory, something that would induce further computational overhead.

Mixing keys with negatives. For MoCHi, in Section 4.2 we propose to synthesize s' synthetic hard negative features for each query, by mixing its feature with a randomly chosen feature from the hardest

[4] https://github.com/SsnL/align_uniform/

negatives in set $\tilde{Q}^N$. We could also create such negatives by mixing the *key* the same way, *i.e*. the synthetic points created this way would be given by $\mathbf{h}''_k = \tilde{\mathbf{h}}''/\|\tilde{\mathbf{h}}''_k\|_2$, where $\tilde{\mathbf{h}}''_k = \beta_k \mathbf{k} + (1-\beta_k)\mathbf{n}_j$, and $\mathbf{n}_j$ is a randomly chosen negative feature from $\tilde{Q}^N$, while $\beta_k \in (0, 0.5)$ is a randomly chosen mixing coefficient for the key. Ablations showed that this yields at best performance as good as mixing with the query, but on average about 0.1-0.2% lower.

Weighted contributions for the logits of $\mathbf{h}'$. We also tried weighing the contributions of the MoCHi samples according to the percentage of the query they have. That is, the logits of each hard negative $\mathbf{h}'_k$ was scaled by β_k to reflect how "negative" this point is. This weighing scheme also resulted in slightly inferior results.

Sampling negatives non-uniformly. We also experimented when sampling negatives with a probability defined over a function of the logit values. That is, we defined a probability function by adding a softmax on top of the top N negatives, with a τ' hyper-parameter. Although we would want to further investigate and thoroughly ablate this approach, early experiments showed that the hard negatives created this way are so hard that linear classification performance decreases by a lot (10-30% for different values of τ').

Fixing the *percentage* of hard negatives in the top-k logits. In an effort to reduce our hyperparameters, we run preliminary experiments on a variant where instead of s and s', we synthesize hard negatives sequentially for each query by alternate between the two mixing methods (*i.e*. mixing two negatives and mixing the query with one negative) up until $X\%$ of the top-N logits correspond to synthetic features. Although encouraging, quantitative results on linear classification were inconclusive; we would however want to further investigate this in the future jointly with curriculum strategies that would decrease this percentage over time.

B.3 Mixing hard negatives vs altering the temperature of the softmax

Another way of making the contrastive loss more or less "peaky" is through the temperature parameter τ of the softmax function; we see from Eq (2) that the gradients of the loss are scaled by $1/\tau$. One would therefore assume that tuning this parameter could effectively tune the hardness and speed of learning. One can see MoCHi as a way of going beyond one generic temperature parameter; we start with the best performing, cross-validated τ and generate different negatives *adapted* to each query, and therefore have adaptive learning for each query that further evolves at each epoch.

Appendix C More experiments and results

C.1 Experimental protocol

In general and unless otherwise stated, we use the default hyperparameters from the official implementation[5] for MoCo-v2. We follow Chen et al. [15] and use a cosine learning rate schedule during self-supervised pre-training. For both pretraining datasets the initial learning rate is set to 0.03, while the batchsize is 128 for ImageNet-100 and 256 for ImageNet-1K. Similar to MoCo-v2 we keep the embedding space dimension to 128. We train on 4 GPU servers. We further want to note here that, because of the computational cost of self-supervised pre-training, 100 epoch pretraining results are computed from the 100th-epoch checkpoints of a 200 epoch run, *i.e*. the cosine learning rate schedule still follows a 200 epoch training. Moreover, our longer (800) epoch runs are by restarting training from the 200 epoch run checkpoint, and switching the total number of epochs to 800, *i.e*., the learning rate jumps back up after epoch 200.

C.1.1 Dataset details

Imagenet. The ImageNet-1K data can be downloaded from this link[6] while the 100 synsets/classes of ImageNet-100 are presented below. For ImageNet-1K the training set is 1.2M images from 1000 categories, while the validation set contains 50 images from each class, *i.e*. 50,000 images in total.

[5] https://github.com/facebookresearch/moco/

[6] http://image-net.org/challenges/LSVRC/2011/

ImageNet-100. ImageNet-100 is a subset of ImageNet-1K that consists of the 100 classes presented right below. It was first used in Tian et al. [69] and recently also used in Shen et al. [62]. The synsets of ImageNet-100 are:

n02869837 n01749939 n02488291 n02107142 n13037406 n02091831 n04517823 n04589890
n03062245 n01773797 n01735189 n07831146 n07753275 n03085013 n04485082 n02105505
n01983481 n02788148 n03530642 n04435653 n02086910 n02859443 n13040303 n03594734
n02085620 n02099849 n01558993 n04493381 n02109047 n04111531 n02877765 n04429376
n02009229 n01978455 n02106550 n01820546 n01692333 n07714571 n02974003 n02114855
n03785016 n03764736 n03775546 n02087046 n07836838 n04099969 n04592741 n03891251
n02701002 n03379051 n02259212 n07715103 n03947888 n04026417 n02326432 n03637318
n01980166 n02113799 n02086240 n03903868 n02483362 n04127249 n02089973 n03017168
n02093428 n02804414 n02396427 n04418357 n02172182 n01729322 n02113978 n03787032
n02089867 n02119022 n03777754 n04238763 n02231487 n03032252 n02138441 n02104029
n03837869 n03494278 n04136333 n03794056 n03492542 n02018207 n04067472 n03930630
n03584829 n02123045 n04229816 n02100583 n03642806 n04336792 n03259280 n02116738
n02108089 n03424325 n01855672 n02090622

PASCAL VOC. For the experiments on PASCAL VOC we use the setup and config files described in MoCo's `detectron2` code[7]. The PASCAL VOC dataset can be downloaded from this link[8]. As mentioned in the main text, we fine-tune a Faster R-CNN [60], R50-C4 on `trainval07+12` and test on `test2007`. Details on the splits can be found here[9] for the 2007 part and here[10] for the 2012 part.

COCO. We similar to PASCAL VOC, we build on top of MoCo's `detectron2` code. We fine-tune all layers end-to-end on the `train2017` set (118k images) and evaluate on `val2017`. The image scale is in [640, 800] pixels during training and is 800 at inference.

C.2 More ablations and results on ImageNet-100

Table 3 presents a superset of the ablations presented in the main text. Please note that most of the results here are *for a single run*. Only results that explicitly present standard deviation were averaged over multiple runs. Some further observations from the extended ablations table:

- From the 100 epoch run results, we see that the gains over MoCo-v2 get larger with longer training. When training longer (see line for 800 epoch pre-training), we see that MoCHi keeps getting a lot stronger, and actually seems to really close the gap even to the supervised case.
- Looking at the smaller queue ablation, we see that MoCHi can achieve with K=4k performance equal to MoCo-v2 with K=16k.
- From the runs using "class oracle" (bottom section), *i.e*. when simply discarding false negatives from the queue, we see that MoCHi comes really close to the supervised case, showing the power of contrastive learning with hard negatives also when labels are present.

C.3 Results for ImageNet-1K

In Table 4 we present a superset of the results presented in the main text, for linear classification on ImageNet-1K and PASCAL VOC. We see that, for 200 epoch training performance still remains strong even when $N = 64$, while the same stands for $N = 128$ when training for 100 epochs. We also added a couple of recent and concurrent methods in the table, *e.g*. PCL [44], or the clustering approach of [10]. Both unfortunately use a different setup for PASCAL VOC and their VOC results are not directly comparable. We see however that our performance for linear classification on ImageNet-1K is higher, despite the fact that both methods take into account the class label-based *clusters* that do exist in ImageNet-1K.

[7] https://github.com/facebookresearch/moco/tree/master/detection
[8] http://host.robots.ox.ac.uk/pascal/VOC/
[9] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/dbstats.html
[10] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/dbstats.html

Oracle run for MoCHi. We also present here a (single) run for MoCHi with a class oracle, when training on ImageNet-1K for 1000 epochs. From this very preliminary result we verify that discarding false negatives leads to significantly higher linear classification performance for ImageNet-1K, the training dataset, while at the same time state-of-the-art transfer learning performance on PASCAL VOC is preserved.

Appendix D An extended related and concurrent works section

Although self-supervised learning has been gaining traction for a few years, 2020 is undoubtedly the year when the number of self-supervised learning papers and pre-prints practically exploded. Due to space constraints, it is hard to properly reference all recent related works in the area in Section 2 of the main text. What is more, a large number of concurrent works on contrastive self-supervised learning came out after the first submission of this paper. We therefore present here an extended related work section that complements that Section (works mentioned and discussed there are not copied also here), a section that further catalogues a large number of concurrent works.

Following the success of contrastive self-supervised learning, a number of more theoretical studies have very recently emerged, trying to understand the underlying mechanism that make it work so well [73, 79, 64, 43, 71], while Mutual Information theory has been the basis and inspiration for a number such studies and self-supervised learning algorithms during the last years, *e.g*. [34, 74, 83, 6, 22, 33]. Building on top of SimCLR [12], the Relational Reasoning approach [56] adds a reasoning module after forming positive and negative pairs; the features for each pair are concatenated and passed through an MLP that predicts a relation score. In RoCL [38], the authors question the need for class labels for adversarially robust training, and present a self-supervised contrastive learning framework that significantly improved robustness. Building on ideas from [47] where objectives are optimized *locally*, in LoCo [86] the authors propose to locally train overlapping blocks, effectively closing the performance gap between local learning and end-to-end contrastive learning algorithms.

Self-supervised learning based on sequential and multimodal visual data. A number of earlier works that learn representations from videos utilized the sequential nature of the temporal dimension, *e.g*. future frame prediction and reconstruction [65], shuffling and then predicting or verifying the order of frames or clips [52, 42, 87], predicting the "arrow of time" [81], pace [78] or predicting the "odd" element [23] from a set of clips. Recently, contrastive, memory-based self-supervised learning methods were extended to video representation learning [29, 33, 68, 72]. In an interesting recent study, Purushwalkam and Gupta [58] study the robustness of contrastive self-supervised learning methods like MoCo [32] and PIRL [51] and saw that despite the fact that they learn occlusion-invariant representations, they fail to capture viewpoint and category instance invariance. To remedy that, they present an approach that leverages unstructured videos and leads to higher viewpoint invariance and higher performance for downstream tasks. Another noteworthy paper that learns visual representations in a self-supervised way from video is the work of Emin Orhan *et al*. [55], that utilized an egocentric video dataset recorded from the perspective of several young children and demonstrated the emergence of high-level semantic information.

A number of works exploit the audio-visual nature of video [41, 1, 57, 16, 3] to learn visual representation, *e.g*. via learning intra-modality synchronization. Apart from audio, other methods have used use automatic extracted text, *e.g*. from speech transcripts [67, 66, 49] or surrounding text [26].

Clustering losses. A number of recent works explore representation learning together with clustering losses imposed on the unlabeled dataset they learn on. Although some care about the actual clustering performance on the training dataset [24, 59, 96], others further use the clustering losses as means for learning representations that generalize [9–11, 92, 90, 5]. Following the recent success of contrastive learning approaches, very recently a number of methods try to get the best of both worlds by combining contrastive and clustering losses. Methods like local aggregation [98], Prototypical Contrastive learning [44], Deep Robust Clustering [96], or SwAV [11] are able to not only create transferable representations, but also are able to reach linear classification accuracy on the training set that is not very far from the supervised case. In a very recent work, Wei *et al*. [80] introduce a consistency regularization method on top of instance discrimination, where they encourage the the similarities of the query and the negatives to match those of the key and the negatives, *i.e*. treating them as a soft distribution over pseudo-labels.

Focusing on the positive pair. Works like SimCLR [12], MoCo-v2 [15] and Tian *et al*. [70] make it clear that for contrastive self-supervised learning, selecting a challenging and diverse set of image transformations can highly boost the quality of representations. Recently, papers like SwAV [11, 88] demonstrated that even higher gains can be achieved by using multiple augmentations. In a very interesting concurrent work, Cai et al. [7] propose a framework where key generation is probabilistic and enables learning with infinite positive pairs in theory, while showing strong quantitative gains.

Very recently the BYOL [28] showed that one can learn transferable visual representations via bootstrapping representations and *without* negative pairs. Recent manuscripts [71] have shown that batch normalization might play an important role when learning without negatives, preventing mode collapse and helping spread the resulting features in the embedding space. Learning without negative was also further very recently explored in [14] where they hypothesize that a stop-gradient operation plays an essential role in preventing collapsing. BYOL makes a number of modifications over SimCLR [12], *e.g*. the addition of a target network whose parameter update is lagging similar to MoCo [32] and a predictor MLP. They further use a different optimizer (LARS) and overall report transfer learning results after 1000 epochs with a batchsize of 4096, a setup that is almost impossible to reproduce (the authors claim training takes about 8 hours on 512 Cloud TPU v3 cores). It is hard to directly compare MoCHi to BYOL, as BYOL does not report transfer learning results for the commonly used setup, *i.e*. after 200 epochs of pre-training. We argue that by employing hard negatives, MoCHi can learn strong transferable representations faster than BYOL.

Synthesizing for supervised metric learning. Recently, synthesizing negatives was explored in metric learning literature [20, 95, 39]. Works like [20, 95] use generators to synthesize negatives in a supervised scenario over common metric learning losses. Apart from not requiring labels, in our case we focus on a specific contrastive loss and exploit its *memory* component. What is more, and unlike [20, 95], we do not require a generator, *i.e.* have no extra parameters or loss terms that need to be optimized. We discuss the relations and differences between MoCHi and the closely related Embedding Expansion [39] method in the related works Section of the main paper.

The MoCHi oracle and supervised contrastive learning. A number of recent approaches have explored the connections between supervised and contrastive learning [37, 94, 46]. Very recently, Khosla et al. [37] show that training a contrastive loss in a supervised way can lead to improvements even over the ubiquitous cross-entropy loss. Although definitely not the focus and merely a byproduct of the class oracle analysis of this paper, we also show here that MoCo and MoCHi can successfully perform supervised learning for classification, by simply discarding same-class negatives. This is something that is further utilized in [94]. For MoCHi, we can further ensure that all *hard negatives* come from other classes. In the bottom section of Table 3 we see that for ImageNet-100, the gap between the cross-entropy and contrastive losses closes more and more with longer contrastive training with harder negatives. An oracle run is also shown in Table 4 for ImageNet-1K after 1000 epochs of training. We see that MoCHi decreases the performance gap to the supervised for linear classification on ImageNet-1K, and performs much better than the supervised pre-training model for object detection on PASCAL. We want to note here that MoCHi oracle experiments on ImageNet-1K are very preliminary, and we leave further explorations on supervised contrastive learning with MoCHi as future work.

Method	Top1 % ($\pm\sigma$)	diff (%)
100 epoch training		
MoCo-v2 [15]*	73.0	
+ MoCHi (1024, 1024, 128)	73.6	↑0.6
+ MoCHi (1024, 128, 256)	73.7	↑**0.7**
200 epoch training		
MoCo [32]	73.4	
MoCo + iMix [62]	74.2‡	↑0.8
CMC [69]	75.7	
CMC + iMix [62]	75.9‡	↑0.2
MoCo [32]* ($t = 0.07$)	74.0	
MoCo [32]* ($t = 0.2$)	75.9	
MoCo-v2 [15]*	78.0 ($\pm$0.2)	
+ MoCHi (16384, 1024, 0)	78.1	↑0.1
+ MoCHi (16384, 0, 1024)	78.5	↑0.4
+ MoCHi (16384, 0, 256)	78.7	↑0.7
+ MoCHi (2048, 1024, 0)	78.2	↑0.2
+ MoCHi (2048, 512, 0)	78.5	↑0.5
+ MoCHi (2048, 512, 256)	78.4	↑0.4
+ MoCHi (1024, 1024, 0)	78.8	↑0.8
+ MoCHi (1024, 1024, 128)	**79.0** ($\pm$0.4)	↑**1.0**
+ MoCHi (1024, 512, 0)	78.9	↑0.9
+ MoCHi (1024, 0, 512)	79.0	↑1.0
+ MoCHi (1024, 256, 512)	**79.0** ($\pm$0.4)	↑**1.0**
+ MoCHi (1024, 128, 256)	**78.9** ($\pm$0.5)	↑**0.9**
+ MoCHi (1024, 128, 0)	78.8	↑0.8
+ MoCHi (1024, 0, 128)	78.7	↑0.7
+ MoCHi (1024, 0, 256)	78.9	↑0.9
+ MoCHi (1024, 0, 512)	79.0	↑1.0
+ MoCHi (512, 128, 0)	78.4	↑0.4
+ MoCHi (512, 512, 0)	78.2	↑0.2
+ MoCHi (512, 128, 128)	78.6	↑0.6
+ MoCHi (256, 256, 0)	78.1	↑0.1
+ MoCHi (256, 512, 0)	77.7	↓0.3
+ MoCHi (128, 128, 0)	77.7	↓0.3
+ MoCHi (64, 6 4, 0)	77.5	↓0.5
$K = 4096$		
MoCo-v2 [15]*	77.5	
+ MoCHi (1024, 1024, 128)	78.0	↑0.5
$K = 1024$		
MoCo-v2 [15]*	76.0	
+ MoCHi (1024, 1024, 128)	77.0	↑1.0
+ MoCHi (1024, 1024, 128) (all queue)	73.4	↓2.6
800 epoch training		
MoCo-v2 [15]*	84.1	
MoCo-v2 [15] + MoCHi (1024, 1024, 128)	**84.5**	
Using Class Oracle		
MoCo-v2* (200 epochs)	81.8	
+ MoCHi (1024, 1024, 128) (200 epochs)	82.5	
+ MoCHi (1024, 1024, 128) (400 epochs)	84.2	
+ MoCHi (1024, 1024, 128) (800 epochs)	85.2	
Cross-entropy classification (supervised)	86.2	

Table 3: More MoCHi ablations on ImageNet-100. Rows without a citation denote reproduced results. ‡ denote results from Figure 4 in [62]. Unless standard deviation is explicitly reported, results in this table are for a single run.

Method	IN-1k Top1	VOC 2007 AP_{50}	VOC 2007 AP	VOC 2007 AP_{75}
100 epoch training				
MoCo-v2 [15]*	63.6	80.8 (±0.2)	53.7 (±0.2)	59.1 (±0.3)
+ MoCHi (512, 1024, 512)	63.7	**81.3** (±0.1) (↑**0.6**)	**54.7** (±0.4) (↑**1.1**)	**60.6** (±0.5) (↑**1.6**)
+ MoCHi (256, 512, 0)	63.9	81.1 (±0.1) (↑0.4)	54.3 (±0.3) (↑0.7)	60.2 (±0.1) (↑1.2)
+ MoCHi (256, 512, 256)	63.7	**81.3** (±0.1) (↑**0.6**)	**54.6** (±0.3) (↑**1.0**)	**60.7** (±0.8) (↑**1.7**)
+ MoCHi (128, 1024, 512)	63.4	81.1 (±0.1) (↑0.4)	**54.7** (±0.3) (↑**1.1**)	**60.9** (±0.1) (↑**1.9**)
200 epoch training				
SimCLR [12] (8k batch size, from [15])	66.6			
DeeperCluster [10] (‡‡train only on VOC 2007)	48.4	71.9‡‡		
MoCo + Image Mixture [62]	60.8	76.4		
InstDis [84]†	59.5	80.9	55.2	61.2
MoCo [32]	60.6	81.5	55.9	62.6
SeLa [5]	61.5			
PIRL [51]†	61.7	81.0	55.5	61.3
InterCLR [85]	65.5			
PCL [44] (‡frozen body)	65.9	78.5‡		
PCL v2 [44]	67.6			
MoCo-v2 [15]	67.7	82.4	57.0	63.6
MoCo-v2 + CO2 [80]	68.0	82.7	57.2	64.1
InfoMin Aug. [70]	70.1	82.7	**57.6**	**64.6**
MoCo-v2 [15]*	67.9	82.5 (±0.2)	56.8 (±0.1)	63.3 (±0.4)
+ MoCHi (1024, 256, 128)	68.0	82.3 (±0.2) (↓0.2)	56.8 (±0.1) (0.0)	63.8 (±0.4) (↑0.5)
+ MoCHi (1024, 512, 256)	68.0	82.3 (±0.2) (↓0.2)	56.7 (±0.2) (↓0.1)	63.8 (±0.2) (↑0.5)
+ MoCHi (1024, 0, 512)	67.8	82.7 (±0.1) (↑0.2)	57.0 (±0.1) (↑0.2)	64.0 (±0.2) (↑0.7)
+ MoCHi (512, 1024, 512)	67.6	82.7 (±0.1) (↑0.2)	57.1 (±0.1) (↑0.3)	64.1 (±0.3) (↑0.8)
+ MoCHi (256, 512, 0)	67.7	**82.8** (±0.2) (↑0.3)	57.3 (±0.2) (↑0.5)	64.1 (±0.1) (↑0.8)
+ MoCHi (256, 512, 256)	67.6	82.6 (±0.2) (↑0.1)	57.2 (±0.3) (↑0.4)	64.2 (±0.5) (↑0.9)
+ MoCHi (256, 2048, 2048)	*67.0*	82.5 (±0.1) (0.0)	57.1 (±0.2) (↑0.3)	64.4 (±0.2) (↑1.1)
+ MoCHi (128, 1024, 512)	*66.9*	82.7 (±0.2) (↑0.2)	57.5 (±0.3) (↑0.7)	64.4 (±0.4) (↑1.1)
+ MoCHi (64, 1024, 512)	*66.3*	82.6 (±0.1) (↑0.1)	57.3 (±0.1) (↑0.5)	64.4 (±0.5) (↑1.1)
800 epoch training				
SvAV [11]	*75.3*	82.6	56.1	62.7
MoCo-v2 [15]	71.1	82.5	**57.4**	64.0
MoCo-v2[15]*	69.0	82.7 (±0.1)	56.8 (±0.2)	63.9 (±0.7)
+ MoCHi (128, 1024, 512)	68.7	**83.3** (±0.1) (↑0.6)	57.3 (±0.2) (↑0.5)	**64.2** (±0.4) (↑0.3)
1000 epoch training				
MoCo-v2[15] + MoCHi (512, 1024, 512)	70.6	83.2 (±0.3)	58.4 (±0.2)	65.5 (±0.6)
MoCo-v2[15] + MoCHi (128, 1024, 512)	69.8	**83.4** (±0.2)	**58.7** (±0.1)	**65.9** (±0.2)
Using Class Oracle				
Cross-entropy classification (supervised)	76.1	81.3	53.5	58.8
MoCo-v2 [15] + MoCHi (512, 1024, 512)	72.6	*83.3*	*57.7*	*64.6*

Table 4: Results on ImageNet-1K and PASCAL VOC. Rows that do not report standard deviation correspond to single runs. Wherever standard deviation is reported for the VOC fine-tuning, results are averaged over three runs. For MoCHi runs we also report difference to MoCo-v2 in parenthesis. * denotes reproduced results. † results are copied from [32].