Max-Margin Contrastive Learning

Anshul Shah[1*] **Suvrit Sra**[2] **Rama Chellappa**[1] **Anoop Cherian**[3*†]

[1]Johns Hopkins University, Baltimore, MD
[2]Massachusetts Institute of Technology, Cambridge, MA
[3]Mitsubishi Electric Research Labs, Cambridge, MA
{ashah95, rchella4}@jhu.edu suvrit@mit.edu cherian@merl.com

Abstract

Standard contrastive learning approaches usually require a large number of negatives for effective unsupervised learning and often exhibit slow convergence. We suspect this behavior is due to the suboptimal selection of negatives used for offering contrast to the positives. We counter this difficulty by taking inspiration from support vector machines (SVMs) to present max-margin contrastive learning (MMCL). Our approach selects negatives as the sparse support vectors obtained via a quadratic optimization problem, and contrastiveness is enforced by maximizing the decision margin. As SVM optimization can be computationally demanding, especially in an end-to-end setting, we present simplifications that alleviate the computational burden. We validate our approach on standard vision benchmark datasets, demonstrating better performance in unsupervised representation learning over state-of-the-art, while having better empirical convergence properties.

Introduction

Learning effective data representations is crucial to the success of any machine learning model. Recent years have seen a surge in algorithms for unsupervised representation learning that leverage the vast amounts of unlabeled data (Chen et al. 2020a; Gidaris, Singh, and Komodakis 2018; Lee et al. 2017; Zhang et al. 2019; Zhan et al. 2020). In such algorithms, an auxiliary learning objective is typically designed to produce generalizable representations that capture some higher-order properties of the data. The assumption is that such properties could potentially be useful in (supervised) downstream tasks, which may have fewer annotated training samples. For example, in (Noroozi and Favaro 2016; Santa Cruz et al. 2018), the pre-text task is to solve patch jigsaw puzzles, so that the representations learned could potentially capture the natural semantic structure of images. Other popular auxiliary objectives include video frame prediction (Oord, Li, and Vinyals 2018), image coloring (Zhang, Isola, and Efros 2016), and deep clustering (Caron et al. 2018), to name a few.

Among the auxiliary objectives typically used for representation learning, one that has gained significant momentum recently is that of contrastive learning, which is a variant of the standard noise-constrastive estimation (NCE) (Gutmann and Hyvärinen 2010) procedure. In NCE, the goal is to learn

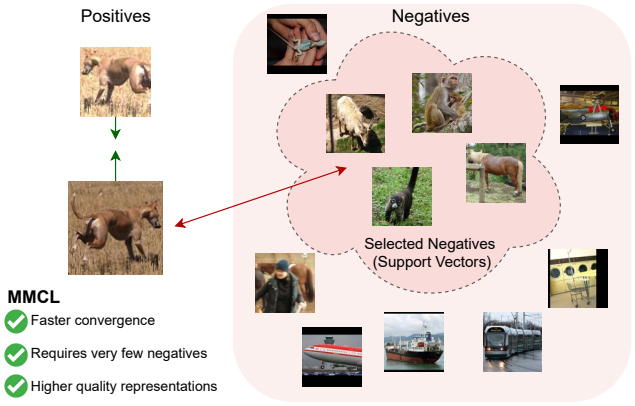

Figure 1: An illustration of our *Max-Margin Contrastive Learning* framework. For every positive example, we compute a weighted subset of (hard) negatives via computing a discriminative hyperplane by solving an SVM objective. This hyperplane is then used in learning to maximize the similarity between the representations of the positives and minimize the similarity between the representations of the positives against the negatives. The negatives in the figure are actual ones selected by our scheme for the respective positive.

data distributions by classifying the unlabeled data against random noise. However, recently developed contrastive learning methods learn representations by designing objectives that capture data invariances. Specifically, instead of using random noise as in NCE, these methods transform data samples to *sets of samples*, each set consisting of transformed variants of a sample, and the auxiliary task is to classify one set (positives) against the rest (negatives). Surprisingly, even by using simple data transformations, such as color jittering, image cropping, or rotations, these methods are able to learn superior and generalizable representations, sometimes even outperforming supervised learning algorithms in downstream tasks (e.g., CMC (Tian, Krishnan, and Isola 2020), MoCo (Chen et al. 2020c; He et al. 2020), SimCLR (Chen et al. 2020a), and BYOL (Grill et al. 2020)).

Typically, contrastive learning methods use the NCE-loss for the learning objective, which is usually a logistic classifier separating the positives from the negatives. However, as is often found in NCE algorithms, the negatives should be *close* in distribution to the positives for the learned representations

*Equal Contribution.
†Corresponding Author.

to be useful – a criteria that often demands a large number of negatives in practice (e.g., 16K in SimCLR (Chen et al. 2020a)). Further, standard contrastive learning approaches make the implicit assumption that the positives and negatives belong to distinct classes in the downstream task (Arora et al. 2019). This requirement is hard to enforce in an unsupervised training regime and defying this assumption may hurt the downstream performance due to beneficial discriminative cues being ignored.

In this paper, we explore alternative formulations for contrastive learning beyond the standard logistic classifier. Rather than contrasting the positive samples against all the negatives in a batch, our key insight is to design an objective that: (i) selects a suitable subset of negatives to be contrasted against, and (ii) provides a means to relax the effect of false negatives on the learned representations. Fig. 1 presents an overview of the idea. A natural objective in this regard is the classical support vector machine (SVM), which produces a discriminative hyperplane with the maximum margin separating the positives from the negatives. Inspired by SVMs, we propose a novel objective, *max-margin contrastive learning* (MMCL), to learn data representations that maximizes the SVM decision margin. MMCL brings in several benefits to representation learning. For example, the *kernel trick* allows for the use of rich non-linear embeddings that could capture desirable data similarities. Further, the decision margin is directly related to the support vectors, which form a weighted data subset. The ability to use slack variables within the SVM formulation allows for a natural control of the influence of false negatives on the representation learning setup.

A straightforward use of the MMCL objective could be practically challenging. This is because SVMs involve solving a constrained quadratic optimization problem, solving which exactly could dramatically increase the training time when used within standard deep learning models. To this end, inspired by coordinate descent algorithms, we propose a novel reformulation of the SVM objective using the assumptions typically used in contrastive learning setups. Specifically, we propose to use a single positive data sample to train the SVM against the negatives – a situation for which efficient approximate solutions can be obtained for the discriminative hyperplane. Once the hyperplane is obtained, we propose to use it for representation learning. Thus, we formulate an objective that uses this learned hyperplane to maximize the classification margin between the remaining positives and the negatives. To demonstrate the empirical benefits of our approach to unsupervised learning, we replace the logistic classifier from prior contrastive learning algorithms with the proposed MMCL objectives. We present experiments on standard benchmark datasets; our results reveal that using our max-margin objective leads to faster convergence and needs far fewer negatives than prior approaches and produces representations that are better generalizable to several downstream tasks, including transfer learning for many-shot recognition, few-shot recognition, and surface normal estimation.

Below, we summarize the key contributions of this work:

- We propose a novel contrastive learning formulation using SVMs, dubbed *max-margin contrastive learning*.
- We present a novel simplification of the SVM objective using the problem setup commonly used in contrastive learning – this simplification allows deriving efficient approximations for the decision hyperplane.
- We explore two approximate solvers for the SVM hyperplane: (i) using projected gradient descent and (ii) closed-form using truncated least squares.
- We present experiments on standard computer vision datasets such as ImageNet-1k, ImageNet-100, STL-10, CIFAR-100, and UCF101, demonstrating superior performances against state of the art, while requiring only smaller negative batches. Further, on a wide variety of transfer learning tasks, our pre-trained model shows better generalizability than competing approaches.

Related Works

While the key ideas in contrastive learning are classical (Becker and Hinton 1992; Gutmann and Hyvärinen 2010; Hadsell, Chopra, and LeCun 2006), it has recently become very popular due to its applications in self-supervised learning. Arguably, objectives based on contrastive learning have outperformed several hand-designed pre-text tasks (Doersch, Gupta, and Efros 2015; Gidaris, Singh, and Komodakis 2018; Larsson, Maire, and Shakhnarovich 2016; Noroozi and Favaro 2016; Zhang, Isola, and Efros 2016). Apart from visual representation learning, the idea of contrastive learning is quickly proliferating into several other subdomains in machine learning, including video understanding (Han, Xie, and Zisserman 2020), graph representation learning (You et al. 2020; Sun et al. 2020), natural language processing (Logeswaran and Lee 2018), and learning audio representations (Saeed, Grangier, and Zeghidour 2021).

In contrastive predictive coding (Oord, Li, and Vinyals 2018), which is one of the first works to apply contrastive learning for self-supervised learning, the noise-contrastive loss was re-targeted for representation learning via the pretext task of future prediction in sequences. It is often empirically seen that the quality of the negatives to be contrasted against has a strong influence on the effectiveness of the representation learned. To this end, for visual representation learning tasks, SimCLR (Chen et al. 2020a,b) proposed a framework that uses a bank of augmentations to generate positives and negatives. As the number of negatives play a crucial role in NCE, many approaches also make use of a memory bank (Chen et al. 2020c; He et al. 2020; Misra and Maaten 2020; Zhuang, Zhai, and Yamins 2019) to enable efficient bookkeeping of the large batches of negatives. Other contrastive learning objectives include: clustering (Caron et al. 2018, 2020; Li et al. 2020a), predicting the representations of augmented views (Grill et al. 2020), and learning invariances (Tian, Krishnan, and Isola 2020; Xiao et al. 2020). The lack of access to class labels in contrastive learning can lead to incorrect learning; e.g., due to false negatives. Recent works have attempted to tackle this issue via avoiding *sampling bias* (Chuang et al. 2020) and adjusting the contrastive loss for the impact of false negatives (Robinson et al. 2021; Huynh et al. 2022; Kalantidis et al. 2020; Iscen et al. 2018). In comparison to these methods that make adjustments to the

NCE loss, we propose an alternative way to view contrastive learning through the lens of max-margin methods using support vector machines; allowing for an amalgamation of the rich literature of SVMs with modern deep unsupervised representation learning approaches.

A key idea in our setup is to view the support vectors as hard negatives for contrastive learning via maximizing the decision margin. Conceptually, this idea is reminiscent of hard-negative mining used in classical supervised learning setups, such as deformable parts models (Felzenszwalb et al. 2009), triplet-based losses (Schroff, Kalenichenko, and Philbin 2015), and stochastic negative mining approaches (Reddi et al. 2019). However, different from these methods, we explore self-supervised losses in this paper, which require novel reformulations of max-margin objectives for making the setup computationally tractable. Our proposed approximations to MMCL result in a one-point-against-all SVM classifier, which is similar to exemplar-SVMs (Malisiewicz, Gupta, and Efros 2011); however rather than learning a bank of classifiers for specific tasks, our objective is to learn embeddings that are generalizable and useful for other tasks.

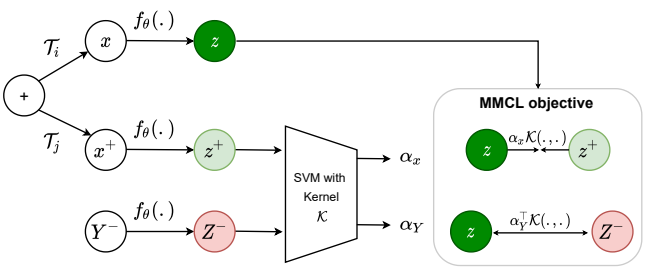

Figure 2: An illustration of our MMCL approach. Given a positive point (+) and a set of negatives Y^-, MMCL learns the parameters θ of a backbone network f_θ via extracting features z^+ and Z^- using a view x^+ of the positive $+$ and the negatives Y^-, respectively. These features are then used in an SVM with an RKHS kernel $\mathcal{K}$ to find a decision hyperplane parameterized by α_x and $\boldsymbol{\alpha}_Y$. Next, MMCL uses the remaining positive views z maximizing the similarity between z and z^+, while minimizing the similarity between z and Z^-, thereby achieving contrastiveness. This ensuing MMCL loss is then backpropagated through the pipeline, thereby learning θ, which is the goal.

Preliminaries

In this section, we review our notation and visit the principles of contrastive learning, support vector machines, and their potential connections, that will set the stage for presenting our approach. We use lower-case for single entities (such as x), and upper-case (e.g., X) for matrices (synonymous with a collection of entities). We use lower-case bold-font (e.g., $\boldsymbol{z}$) for vectors. For a function, say f, defined on vectors, we sometimes overload it as $f(X)$, by which we mean applying f to each entity in X.

Contrastive Learning

Suppose $\mathcal{D} = \{x_i\}_{i=1}^N$ is a given unlabeled dataset, where each $x_i \in \mathbb{R}^d$. Let $\mathcal{T} : \mathbb{R}^d \to \mathbb{R}^d$ denote a random cascade of data transformation maps (e.g., random image crops and rotations). Standard contrastive learning methods use $\mathcal{T}$ to augment $\mathcal{D}$, thereby producing sets of data points $\mathcal{D}' = \{X_1, X_2, \cdots, X_N\}$, where each X is a (potentially infinite) set of transformed data samples obtained via randomly applying $\mathcal{T}$ on each x, i.e., $X = \{\mathcal{T}(x)\}$. The task of representation learning then amounts to minimizing an objective that maximizes the similarity between points from within a set against data points from other sets – essentially learning the data manifold in some representation space, with the hope that such representations are useful in subsequent tasks.

Suppose $\boldsymbol{f}_\theta : \mathbb{R}^d \to \mathbb{R}^{d'}$ denote a function mapping a data point x to its representation, i.e., $\boldsymbol{f}_\theta(x)$. Then, inspired by noise-contrastive estimation (Gutmann and Hyvärinen 2010), contrastive learning methods learn the function $\boldsymbol{f}_\theta$ via minimizing the empirical logistic loss (with respect to θ):

$$-\sum_{X \in B} \log \frac{g(\boldsymbol{f}_\theta(x), \boldsymbol{f}_\theta(x^+))}{g(\boldsymbol{f}_\theta(x), \boldsymbol{f}_\theta(x^+)) + \sum_{x^- \in B \setminus X} g(\boldsymbol{f}_\theta(x), \boldsymbol{f}_\theta(x^-))}, \tag{1}$$

over batches $B \subset \mathcal{D}'$ with positives $\{x, x^+\} \subset X \in B$, negatives $x^- \in X'$, where $X' \in B \backslash X$, and using a suitable similarity function g (e.g., a learnable projection-head followed by an exponentiated-cosine distance as in SimCLR (Chen et al. 2020a)). As alluded to earlier, the contrastive learning loss in (1) poses several challenges from a representation learning perspective. For example, in the absence of any form of supervision, this learning objective needs to derive the training signals from the (thus far learned) representations of the negative pairs, which could be very noisy; thereby requiring very large negative batches. However, having such large batches increases the chances of *class collisions*, i.e., positives and negatives belonging to the same class in a subsequent downstream task; such collisions have been shown to be detrimental (Arora et al. 2019). As alluded to earlier, unlike approaches that attempt to circumvent this issue, such as (Huynh et al. 2022; Robinson et al. 2021; Chuang et al. 2020), we seek to explore alternative contrastive learning objectives that are less sensitive to issues discussed above using formulations that maximize the discriminative margin between the positives and the negatives.

Note that instead of the InfoNCE loss, as in (1), for contrasting the positives from the negatives, an alternative is perhaps the hinge loss (Arora et al. 2019; Chen et al. 2020a), that minimizes (with respect to θ):

$$\sum_{x, x^+, x^-} \big[\, t - \mathrm{sim}(\boldsymbol{f}_\theta(x), \boldsymbol{f}_\theta(x^+)) + \mathrm{sim}(\boldsymbol{f}_\theta(x), \boldsymbol{f}_\theta(x^-)) \big]_+ ,$$

where $[\,.\,]_+ = \max(0,\ .)$ denotes the hinge loss and t is a margin hyperparameter that must be tuned manually. Our proposed scheme avoids the need for this hyperparameter as the margin is an objective of the optimization.

Support Vector Machines

Given two sets X^+ and X^- with labels $y_x = 1$, if $x \in X^+$ and -1 otherwise, the soft-margin SVM solves the objective:

$$\min_{\boldsymbol{w},b,\xi\geq 0} \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 + C\sum_x \xi_x$$
$$\text{s. t. } y_x(\boldsymbol{w}^\top x + b) \geq 1 - \xi_x, \forall x \in X^+ \cup X^-, \quad (2)$$

where $\boldsymbol{w}$ denotes the discriminative hyperplane separating the two classes, b is a bias, and ξ_x is a per-data-point non-negative slack with a penalty C that balances between misclassification of *hard* points and maximizing the decision margin. It is well-known that $1/\left\|\boldsymbol{w}\right\|$ captures the margin between the positives and the negatives, and thus the objective in (2) attempts to find the hyperplane $\boldsymbol{w}$ that maximizes this margin. The Lagrangian dual of (2) is given by:

$$\min_{0\leq\boldsymbol{\alpha}\leq C,\boldsymbol{\alpha}^\top\boldsymbol{y}=0} \tfrac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{K}(X^+,X^-)\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top\mathbf{1}, \quad (3)$$

where $\boldsymbol{K} \in S_{++}^{|X^+\cup X^-|}$ denotes a symmetric positive definite kernel matrix, the ij-th element of which is given by: $\boldsymbol{K}_{ij} = y_{x_i}y_{x_j}\mathcal{K}(x_i,x_j)$ for some suitable RKHS kernel $\mathcal{K}$ and $x_i, x_j \in X^+ \cup X^-$. As the formulations in (2) and (3) are convex, a solution $\boldsymbol{\alpha}$ to (3) provides the exact decision hyperplane for (2) and is given by:

$$\boldsymbol{w}(.) = \sum_{x\in X^+\cup X^-} \alpha_x y_x \mathcal{K}(x,.). \quad (4)$$

As the bias term b in (2) is not essential for the details to follow, we will not need the exact form of this term and will use $\boldsymbol{w}(.)$ to refer to the decision hyperplane.

Proposed Method

In this section, we connect the approaches described above deriving our MMCL formulation. An overview of our approach is illustrated in Figure 2.

Contrastive Learning Meets SVMs

The advantages of SVM listed in the last section may seem worthwhile from a contrastive representation learning perspective, and suggest directly using SVM instead of the logistic classifier in (1). Formally, using a soft-constraint variant of (2) with a margin t, the optimization problem in (1) can be re-written as:

$$\min_\theta \sum_{B\subset D'}\sum_{X\in B} \min_{\boldsymbol{w}_X}\Big(\tfrac{1}{2}\left\|\boldsymbol{w}_X\right\|^2 + \left[t - \langle \boldsymbol{w}_X, \boldsymbol{f}_\theta(X)\rangle\right]_+ + \sum_{X^-\in B\setminus X}\left[t + \langle \boldsymbol{w}_X, \boldsymbol{f}_\theta(X^-)\rangle\right]_+\Big), \quad (5)$$

where X, X^- denote the sets of positives and negatives respectively, and w_X captures a max-margin hyperplane separating them.[1] The inner optimization over *each* w_X is what translates into training an SVM. We augment this inner optimization problem in two ways: (i) by including slack variables to model a soft-margin (as in (2)), which results in a hyperparameter C; and (ii) by permitting an additional non-linear feature map ϕ so that we may use $\phi(\boldsymbol{f}_\theta)$ (as in (3)) in (5). Using these changes, a contrastive learning formulation via maximizing the SVM classification margin may be derived (by rewriting (5)) as:

$$\min_\theta \mathcal{L}(\theta) := \sum_{B\subset\mathcal{D}'}\sum_{X\in B} {\boldsymbol{\alpha}_X^*}^\top \boldsymbol{K}\left(\boldsymbol{f}_\theta(X), \boldsymbol{f}_\theta(B\setminus X)\right)\boldsymbol{\alpha}_X^*, \quad (6)$$

$$\text{s.t.} \quad \boldsymbol{\alpha}_X^* = \underset{\substack{0\leq\boldsymbol{\alpha}\leq C,\\ \boldsymbol{\alpha}^\top\boldsymbol{y}=0}}{\arg\min}\ \tfrac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{K}(\boldsymbol{f}_\theta(X),\boldsymbol{f}_\theta(B\setminus X))\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top\mathbf{1}, \quad (7)$$

where $\boldsymbol{K}(Z^+,Z^-) = \begin{bmatrix} \mathcal{K}(Z^+,Z^+), -\mathcal{K}(Z^+,Z^-) \\ -\mathcal{K}(Z^-,Z^+), \mathcal{K}(Z^-,Z^-)\end{bmatrix}$ is a kernel matrix induced by the RKHS kernel $\mathcal{K}(\boldsymbol{z},\boldsymbol{z}') = \langle\phi(\boldsymbol{z}),\phi(\boldsymbol{z}')\rangle$. While SVMs have been widely studied in the machine learning literature (Smola and Schölkopf 1998; Cortes and Vapnik 1995), our idea of linking the fields of SVMs and Contrastive Learning has not been explored before.

In (7), we use the so-far trained $\boldsymbol{f}_\theta$ to produce $\boldsymbol{\alpha}_X^*$ defining the decision margin, which is then used in (6) to update θ while striving to maximize the margin; doing so, pushes the support vectors from the positive and negative classes away from each other. Unfortunately, despite its intuitive simplicity, the formulation (6)-(7) is impractical to use directly. Indeed, it is a challenging bilevel optimization problem (Gould et al. 2016; Amos and Kolter 2017; Wang et al. 2018), and if we use an iterative SVM solver for the lower problem (7) within a deep learning framework, it can incur significant slowdown.

Remarks. There are several interesting aspects of the SVM solution that are perhaps beneficial from a contrastive learning perspective: (i) the dual solution $\boldsymbol{\alpha}$ is usually sparse[2], and its active dimensions can be used to identify data points that are the support vectors defining the decision margin, (ii) the slack regularization controls the misclassification rate, and allows tuning the performance against the class collisions, similar to (Chuang et al. 2020), (iii) the dimensions of $\boldsymbol{\alpha}_X$ are equal to C for misclassified points, which are perhaps hard or false negatives, and thus our formulation allows for identifying these points and mitigate their effects, and (iv) the use of the kernel function provides rich RKHS similarities at our disposal allowing to use, for example, novel structures within the learned representations (e.g., trees, graphs, etc.).

Max-Margin Contrastive Learning

The primary method for solving (6) is stochastic gradient descent (SGD), which computes stochastic gradients over the batches $B \subset \mathcal{D}'$ via backpropagation while iteratively updating θ. However, as has been previously observed for bilevel optimization (Amos and Kolter 2017; Gould et al. 2016), even obtaining a single stochastic gradient requires solving the lower problem (7) exactly, which is impractical. Our key idea to overcome this challenge is to introduce a

[1] Note that $\boldsymbol{f}_\theta(\Lambda)$ we mean applying $\boldsymbol{f}_\theta$ to each item in set Λ.

[2] When the $\mathcal{K}$ is chosen appropriately.

"*sample splitting*" trick inspired by coordinate descent, which helps to reduce the computational burden. Subsequently, we make additional approximations that lead to our final training procedure.

Without loss of generality, assume that X consists of the pair (x, x^+); the same idea applies if we permit multiple such positive pairs in X. Instead of solving (7) using all the "coordinates", we *split* the pair (x, x^+) into two parts: (i) x^+, which is used to perform coordinate descent on (7); and (ii) x, which is used to perform the SGD step for (6). This splitting aligns well with contrastive learning, where often one uses only a pair of positives that must be contrasted against the negatives.

The following proposition states how we perform part (i) of our split to estimate $\boldsymbol{\alpha}_X^*$, which we will henceforth denote as $\boldsymbol{\alpha}_x$ to indicate its dependence on the split sample.

Proposition 1. *Let (x^+, Y^-) be a tuple consisting of a positive point $x^+ \in \mathbb{R}^d$ and a set of n negative points $Y^- \in \mathbb{R}^{d\times n}$. Further, let $\boldsymbol{z}^+ = \boldsymbol{f}_\theta(x^+)$ and $\boldsymbol{Z}^- = \boldsymbol{f}_\theta(Y^-)$. Suppose $k_{xx}, \boldsymbol{k}_{xY}$, and $\boldsymbol{K}_{YY}$ denote $\mathcal{K}(\boldsymbol{z}^+, \boldsymbol{z}^+)$, $\mathcal{K}(\boldsymbol{z}^+, \boldsymbol{Z}^-)$, and $\mathcal{K}(\boldsymbol{Z}^-, \boldsymbol{Z}^-)$, respectively. Consider the SVM decision function for a new point $\boldsymbol{z}$ given by*

$$\boldsymbol{w}(\boldsymbol{z}) = \boldsymbol{\alpha}_x^\top \left(\mathcal{K}(\boldsymbol{z}^+, \boldsymbol{z})\mathbf{1} - \mathcal{K}(\boldsymbol{Z}^-, \boldsymbol{z})\right). \tag{8}$$

Let $\boldsymbol{\Delta} = \mathbf{1}\mathbf{1}^\top + \boldsymbol{K}_{YY} - \boldsymbol{k}_{xY}\mathbf{1}^\top - \mathbf{1}\boldsymbol{k}_{xY}^\top$, and let $P_{[0,C]}$ denote projection onto the interval $[0, C]$. By suitably selecting $\boldsymbol{\alpha}_x$ in (8) *we then obtain the following approximate max-margin solutions:*

(i) (block) coordinate minimization $\boldsymbol{\alpha}_x^{cm} = \arg\min_{0\le\boldsymbol{\alpha}\le C} g(\boldsymbol{\alpha}) := \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{\Delta}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top\mathbf{1}$,

(ii) m-step projected gradient ($\mathrm{MMCL}_{\mathrm{PGD}}$): $\boldsymbol{\alpha}_x^{pg} := \boldsymbol{\alpha}_m = P_{[0,C]}(\boldsymbol{\alpha}_{m-1} - \eta(\boldsymbol{\Delta}\boldsymbol{\alpha}_{m-1} - 2\mathbf{1}))$, for some initial guess $\boldsymbol{\alpha}_0 \in [0, C]^n$, $\eta > 0$ a step-size, and

(iii) greedy truncated least-squares ($\mathrm{MMCL}_{\mathrm{INV}}$): $\boldsymbol{\alpha}_x^{ls} = P_{[0,C]}(2\boldsymbol{\Delta}^{-1}\mathbf{1})$.

The various solutions satisfy $g(\boldsymbol{\Delta}^{-1}1) \le g(\boldsymbol{\alpha}_x^{cm}) \le \min\{g(\boldsymbol{\alpha}_x^{pg}, g(\boldsymbol{\alpha}_x^{ls})\}$. Moreover, $g(\boldsymbol{\alpha}_x^{pg}) - g(\boldsymbol{\alpha}_x^{cm}) = O\left(\exp\left(-m\frac{\lambda_{\min(\boldsymbol{\Delta})}}{\lambda_{\max(\boldsymbol{\Delta})}}\right)(g(\boldsymbol{\alpha}_0) - g(\boldsymbol{\alpha}_x^{cm}))\right)$.

Proof. Choice (i) is obvious. To obtain (ii) and (iii), consider the following dual SVM formulation:

$$\min_{0\le\boldsymbol{\alpha}_Y\le C} \frac{1}{2}\begin{bmatrix}\alpha_x\\ \boldsymbol{\alpha}_Y\end{bmatrix}^\top \begin{bmatrix}k_{xx} & -\boldsymbol{k}_{xY}^\top\\ -\boldsymbol{k}_{xY} & \boldsymbol{K}_{YY}\end{bmatrix}\begin{bmatrix}\alpha_x\\ \boldsymbol{\alpha}_Y\end{bmatrix} - \left[\alpha_x + \boldsymbol{\alpha}_Y^\top\mathbf{1}\right],$$

where $\alpha_x = \boldsymbol{\alpha}_Y^\top\mathbf{1}$. Substituting for α_x, we obtain[3]:

$$\min_{0\le\boldsymbol{\alpha}_Y\le C} g(\boldsymbol{\alpha}_Y) = \tfrac{1}{2}\boldsymbol{\alpha}_Y^\top\boldsymbol{\Delta}\boldsymbol{\alpha}_Y - 2\boldsymbol{\alpha}_Y^\top\mathbf{1}.$$

Setting $\nabla g(\boldsymbol{\alpha}_Y) = 0$, we obtain the unconstrained least-squares solution $2\boldsymbol{\Delta}^{-1}\mathbf{1}$, which we can greedily truncate to lie in the interval $[0, C]$ to obtain (iii). Solution (ii) runs m iterations of projected gradient descent, and hence it also satisfies a linear convergence rate, which rapidly brings it within the optimal solution $\boldsymbol{\alpha}_x^{\mathrm{cm}}$ at the well-known rate depending on the condition number $\lambda_{\max(\boldsymbol{\Delta})}/\lambda_{\min(\boldsymbol{\Delta})}$. □

[3]Note that α_x is the scalar Lagrangian dual associated with the data point $\boldsymbol{z}$ while $\boldsymbol{\alpha}_x$ is the vector of all dual variables associated with the batch B when considering x as the positive.

Algorithm 1: Pseudocode for MMCL

Input: Dataset $\mathcal{D}$, batch size N, encoder $\boldsymbol{f}_\theta$, slack-penalty C, kernel $\mathcal{K}$, augmentation map $\mathcal{T}$
for minibatch $B = \{\boldsymbol{x}_k\}_{k=1}^N \sim \mathcal{D}$ **do**
  `loss` := 0
  for $k = 1, \ldots, N$ **do**
    draw $t_1 \sim \mathcal{T}$, $t_2 \sim \mathcal{T}$
    # get embeddings for positives and negatives
    $\boldsymbol{z}^+ = \boldsymbol{f}_\theta(t_1(\boldsymbol{x}_k))$, $\boldsymbol{z} = \boldsymbol{f}_\theta(t_2(\boldsymbol{x}_k))$
    $\boldsymbol{Z}^- = \boldsymbol{f}_\theta(t_1(B \setminus \boldsymbol{x}_k) \cup t_2(B \setminus \boldsymbol{x}_k))$
    # calculate kernel similarities
    $\boldsymbol{k}_{xY} = \mathcal{K}(\boldsymbol{z}^+, \boldsymbol{Z}^-)$, $\boldsymbol{K}_{YY} = \mathcal{K}(\boldsymbol{Z}^-, \boldsymbol{Z}^-)$
    # Solve SVM
    $\boldsymbol{\Delta} = \mathbf{1}\mathbf{1}^\top + \boldsymbol{K}_{YY} - \boldsymbol{k}_{xY}\mathbf{1}^\top - \mathbf{1}\boldsymbol{k}_{xY}^\top$
    $\boldsymbol{\alpha}_x$ = `svm_solver`$(\boldsymbol{\Delta}, C)$ # using PGD or INV
    # calculate the loss
    `loss` += $\boldsymbol{\alpha}_x^\top\left(\mathcal{K}(\boldsymbol{Z}^-, \boldsymbol{z}) - \mathcal{K}(\boldsymbol{z}^+, \boldsymbol{z})\mathbf{1}\right)$
  end for
  update the model to minimize the loss
end for

Using Prop. 1, we can reformulate the contrastive learning objective in (6) as maximizing the margin in classifying the other part of the split, namely the positive point x, correctly against the negatives. Here, we introduce an additional simplification by rewriting the margin in terms of the separation between x and Y^-, using the decision hyperplane (8). Let $\boldsymbol{\alpha}_x$ denote the solution obtained from Proposition 1 using the positive point x. Then, we rewrite (6) into our proposed *max-margin contrastive learning* objective as:

$$\min_\theta \sum_{\substack{(x,x^+)\sim B\in\mathcal{D}'\\ Y^- = B\setminus(x,x^+)}} \boldsymbol{\alpha}_x^T\left[\mathcal{K}\left(\boldsymbol{f}_\theta(Y^-), \boldsymbol{f}_\theta(x)\right) - \mathbf{1}\mathcal{K}\left(\boldsymbol{f}_\theta(x^+), \boldsymbol{f}_\theta(x)\right)\right]. \tag{9}$$

When optimizing for θ, (9) seeks a representation map $\boldsymbol{f}_\theta$ that improves the similarity between the positives (x, x^+) and the dissimilarity between x and all the points in Y^-, achieving a similar effect as in standard contrastive learning objective in (1), but with the advantage of choosing kernels, selecting the support vectors that matter to the decision margin, as well as finding points that are perhaps hard negatives (those at the upper-bound of the box-constraints), all in one formulation. Note that, using the exact solver (i) in Prop. 1 turned out to be prohibitively expensive in standard contrastive learning pipelines and thus we do not use that variant in our experiments. In Algorithm 1, we provide a pseudocode highlighting the key steps in our approach.

Experiments and Results

In this section, we systematically study the various components in MMCL, as well as compare performances of MMCL-learned representations for their quality via linear evaluation, and their generalizability on transfer learning tasks.

Visual Representation Learning Experiments. We base our experimental setup on the popular SimCLR (Chen et al. 2020a) baseline, which is widely used, especially to evaluate

Method	Many-Shot classification								Few-Shot classification		Surface Normal Est.
	Aircraft	Caltech101	Cars	CIFAR10	CIFAR100	DTD	Flowers	Food	CropDiseases	EuroSat	NYUv2 (Angular error)
Supervised	83.5	91.01	82.61	96.39	82.91	73.30	95.50	84.60	93.09 ± 0.43	88.36 ± 0.44	27.91
InsDis (Wu et al. 2018)	73.38	72.04	61.56	93.32	68.26	63.99	89.51	76.78	91.95 ± 0.44	86.52 ± 0.51	27.35
MoCo (He et al. 2020)	75.61	74.95	65.02	93.89	71.52	65.37	89.45	77.28	92.04 ± 0.43	86.55 ± 0.51	28.63
PCL-v1 (Li et al. 2020a)	74.97	**87.62**	73.24	**96.35**	79.62	70	90.83	78.3	80.74 ± 0.57	75.19 ± 0.67	33.58
PCL-v2 (Li et al. 2020a)	79.37	88.04	71.68	**96.5**	80.26	71.76	92.95	80.34	92.58 ± 0.44	87.94 ± 0.4	28.67
MoCo-v2 (Chen et al. 2020c)†	82.46	82.31	85.1	96.06	72.99	69.41	**95.62**	77.19	90.01 ± 0.48	88.06 ± 0.4	**24.49**
MoCHI (Kalantidis et al. 2020)†	83.03	84.45	85.49	95.68	77.07	70.85	94.8	78.9	91.93 ± 0.46	$\mathbf{88.26 \pm 0.4}$	31.75
Ours	**85.38**	**87.82**	**89.23**	**96.24**	**82.09**	**73.51**	**95.24**	**82.39**	$\mathbf{93.1 \pm 0.45}$	$\mathbf{88.75 \pm 0.4}$	**24.69**

Table 1: Transfer learning results. We transfer an ImageNet-pre-trained model (using MMCL) on a range of downstream tasks and datasets. We compare with models pre-trained using a similar batch size and epochs. Results on competing approaches are taken from (Ericsson, Gouk, and Hospedales 2021). †Models evaluated using publicly available checkpoints.

the effectiveness of the "learning loss" against other factors in a self-supervised algorithm (e.g., data augmentations, use of queues, multiple crops). We use a ResNet50 backbone, followed by a two-layer MLP as the projection-head, followed by unit normalization. We pretrain our models on ImageNet-1K (Deng et al. 2009) using the LARS optimizer (You, Gitman, and Ginsburg 2018) with an initial learning rate of 1.2 for 100 epochs. We also present results on ImageNet-100 (Tian, Krishnan, and Isola 2020) (a subset of ImageNet-1K) and on smaller datasets such as STL-10 (Coates, Ng, and Lee 2011) and CIFAR-100 (Krizhevsky, Hinton et al. 2009), especially for our ablation studies. We pre-train on ImageNet-100 for 200 epochs in our studies, while pre-training for 400 epochs on the smaller datasets. We use the Adam optimizer with a learning rate of 1e-3 as in (Chuang et al. 2020; Robinson et al. 2021). Unless otherwise stated, we use a batch-size of 256 for all ImageNet-1K, CIFAR-100, and STL-10 experiments and 128 for ImageNet-100 experiments. In addition, we also present results on video representation learning using an S3D backbone (Xie et al. 2018) on the UCF-101 (Soomro, Zamir, and Shah 2012) dataset, pre-trained using MMCL for 300 epochs.

Hyperparameters: We mainly use the RBF kernel for the SVM. For CIFAR-100 experiments, we start with a kernel bandwidth $\sigma^2 = 0.02$ and increase it by a factor of 10 at 75 and 125 epochs. For STL-10 experiments, we use a kernel bandwidth $\sigma^2 = 1$. We used $\sigma^2 = 5$ for ImageNet experiments. We set the SVM slack regularization C to 100. For the projected gradient descent optimizer for MMCL, we use a maximum of 1000 steps. Additional details are provided in the Appendix.

Practical Considerations: Here, we note a few important but subtle technicalities that need to be addressed when implementing MMCL. Specifically, we found that backpropagating the gradients through $\boldsymbol{\alpha}_Y$ is prohibitively expensive when using PGD iterations. On the other hand, for the least-squares variant, gradients through $\boldsymbol{\alpha}_Y$ was found to be detrimental. This is perhaps unsurprising, because note that, $\boldsymbol{\alpha}_Y$ term includes the term Δ^{-1}. To improve the decision margin, one needs to make Δ an identity matrix, so that the off-diagonal elements go to zero during optimization, which suggests that the training gradients should reduce the magnitude of these terms. However, on the other hand, as $\boldsymbol{\alpha}_Y$ uses Δ^{-1} one could also maximize the margin by making Δ ill-conditioned, via making the off-diagonal elements going to one. Such a tug-of-war between the gradients can essentially destabilize the training. Thus, we found that avoiding any backpropagation through $\boldsymbol{\alpha}$ is essential for MMCL to learn to produce representations. We also found that using a small regularization $\Delta + \beta I$ ($\beta = 0.1$) is necessary for the learning to begin. This is because, initially the representations can be nearly zero, and thus the kernel may be poorly conditioned.

Experiments on Transfer Learning

Recently, models pretrained using various self-supervised learning approaches have shown impressive performance when transferred to various downstream tasks. In this section, we evaluate MMCL-ImageNet pretrained models on such downstream tasks. For these experiments, we follow the experimental protocol provided in (Ericsson, Gouk, and Hospedales 2021). We evaluate the models in the fine-tuning setting and use the benchmarking scripts provided in (Ericsson, Gouk, and Hospedales 2021) without any modifications.

First, we transfer the MMCL-pretrained backbone model to a collection of many-shot classification datasets used in (Ericsson, Gouk, and Hospedales 2021), namely FGVC Aircraft, Caltech-101, Stanford Cars, CIFAR-10, CIFAR-100, DTD, Oxford Flowers, and Food-101. The setup involves using the pretrained model as the initial checkpoint and attaching a task specific head to the backbone model. The entire network is then finetuned for the downstream task. These datasets vary widely in content and texture compared to ImageNet images. Further, the benchmark datasets include a significant diversity in the number of training images (2K–50K) and the number of classes (10–196). For a fair comparison, we only include results for models which are trained for a comparable number of epochs and batch sizes. For few-shot experiments, we follow the setup described in (Ericsson, Gouk, and Hospedales 2021) for few-shot learning on the Cross-Domain Few-Shot Learning (CD-FSL) benchmark (Guo et al. 2020). We evaluate on Crop-Diseases (Mohanty, Hughes, and Salathé 2016), EuroSAT (Helber et al. 2019) datasets for 5-way 20-shot transfer. Finally, we evaluate performance of our model for the dense prediction task of surface normal estimation on NYUv2 (Silberman et al. 2012) and report the median angular error.

In Table 1, we provide results on the transfer learning experiments. We see that MMCL consistently outperforms the competing self-supervised learning approaches on a wide variety of transfer tasks and across all datasets. Further,

MMCL also outperforms the supervised counterpart on several datasets. These results show that MMCL learns high-quality generalizable features.

Experiments on Linear Evaluation

For these experiments, we freeze the weights of the backbone (ResNet-50), and attach a linear layer as in (Chen et al. 2020a), which is trained using the class labels available with the dataset. We train this linear layer for 100 epochs. Tables 2 and 3 show our results. We see that MMCL-pretrained model outperforms SimCLR by 6.3% on ImageNet-1K using the same number of negatives. We also compare with the recent memory queue-based methods such as MoCo-v2 (Chen et al. 2020c) and MoCHI (Kalantidis et al. 2020), demonstrating competitive performances while using far fewer negatives (510 vs 65536). We also establish a new state of the art on ImageNet-100 outperforming MoCHI by 1.7% using only 510 negatives (0.008x) and without a memory bank.

Variant	Negative Source	Negatives	top-1
MoCo	Memory Queue	16000	75.9
CMC	Memory Queue	16000	75.7
MoCo-v2	Memory Queue	16000	78.0
MoCHI	Memory Queue	16000	79.0
Ours	Batch	254	**80.7**

Table 2: ImageNet-100 linear evaluation.

Variant	Negative Source	Negatives	top-1
SimCLR	Batch	254	57.5
SimCLR	Batch	510	60.62
MoCo-v2	Memory Queue	65536	63.6
MoCHI	Memory Queue	65536	**63.9**
Ours	Batch	510	**63.8**

Table 3: ImageNet-1K linear evaluation.

Experiments on Graph Representation Learning

Recall that our MMCL formulation works by modifying the contrastive learning loss function; and as a result, our approach is generically applicable to a variety of tasks. In this section, we evaluate our approach on learning graph representations using contrastive learning. We experiment with five common graph benchmark datasets MUTAG (Kriege and Mutzel 2012) – a dataset containing mutagenic compounds, DD(Yanardag and Vishwanathan 2015)– a dataset of biochemical molecules, REDDIT-BINARY, REDDIT-M5K, and IMDB-BINARY (Yanardag and Vishwanathan 2015) which are social network datasets. Our experiments use GraphCL (You et al. 2020) – a projection head based on the contrastive learning framework derived from SimCLR while incorporating graph augmentations. For these experiments, we follow the training and evaluation protocols described in (You et al. 2020). Specifically, we use the standard ten-fold cross validation using an SVM and report the average performances and their standard deviations. We use the Adam optimizer for training these models. Table 4 shows the results of using MMCL instead of the NCE loss. We see that adding MMCL is comparable or better than GraphCL for these datasets. On MUTAG, we obtain an absolute improvement of 1.62% over GraphCL. These results demonstrate the effectiveness of our approach in learning better representations. Given that the only change from GraphCL is the underlying objective, the results also show that our approach is general and can easily replace NCE based losses.

Experiments on Video Action Recognition

For this experiment, we use the S3D backbone model (Xie et al. 2018) pre-trained using MMCL on RGB and optical flow images from the UCF-101 dataset. We pre-train the network for 300 epochs, followed by 100 epochs for linear evaluation on the task of action recognition. We report the standard 10-crop test accuracy on split-1, as well as on nearest neighbor retrieval. As seen in Table 5, MMCL outperforms the baseline by 5.65% on RGB and 1.21% on flow in linear evaluation and 12.5% and 5.74% on Retrieval@1, demonstrating the generalizability of our approach to the video domain.

Ablation studies and Analyses

For some of the ablation experiments, we use the smaller datasets: STL-10 and CIFAR-100, and report the readout accuracy calculated using k-NN with k=200 at 200 epochs, besides standard evaluations.

Choice of Kernels: Unlike the traditional NCE objective, our approach naturally allows for the use of kernels to better capture the similarity between the data points. In Table 6, we compare the readout accuracy on CIFAR100 and STL10 for various choices of popular kernels. As is clear from the table, the RBF kernel performs better on both datasets. The best kernel hyperparameters σ, γ were found empirically. We choose the RBF kernel in our subsequent experiments.

Effect of slack: A key benefit of our MMCL formulation is the possibility to use a slack that could potentially control the impact of false or hard negatives. To evaluate this effect, we changed the slack penalty C from 0.01 (i.e., low penalty for misclassification) to $C = \infty$. The results on readout accuracy in Figure 3 shows that C plays a key role in achieving good performance. For example, with $C = 0.01$, it appears that the performance is consistently low for both the datasets, perhaps because the hard negatives are under-weighted. We also find that using a large C may not be beneficial always.

Effect of batch size: We use STL-10 dataset for this experiment and train all models for 400 epochs. We report the Linear evaluation results for this experiment. From Table 7, we see that our model consistently outperforms the SimCLR baseline, while performing much better than other approaches. Indeed, we find that MMCL is about 1-3% better than HCL, which reweights the hard negatives. We also find that MMCL using a batch size of only 128 reaches close to the performance of HCL (Robinson et al. 2021) trained using a batch size of 256, suggesting that the proposed selection of

Method	DD	MUTAG	REDDIT-BIN	REDDIT-M5K	IMDB-BIN
GraphCL	$\mathbf{78.62 \pm 0.40}$	86.80 ± 1.34	89.53 ± 0.84	55.99 ± 0.28	71.14 ± 0.44
Ours	$\mathbf{78.74 \pm 0.30}$	$\mathbf{88.42 \pm 1.33}$	$\mathbf{90.41 \pm 0.60}$	$\mathbf{56.18 \pm 0.29}$	$\mathbf{71.62 \pm 0.28}$

Table 4: Comparison with GraphCL. We compare graph representation learning on five graph benchmark datasets. The compared numbers are obtained from the original paper (You et al. 2020).

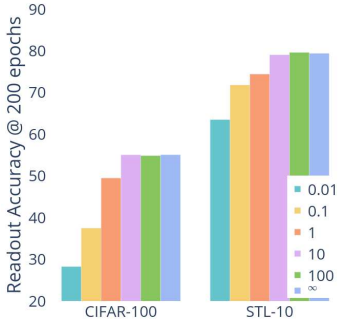

Figure 3: Effect of slack penalty C.

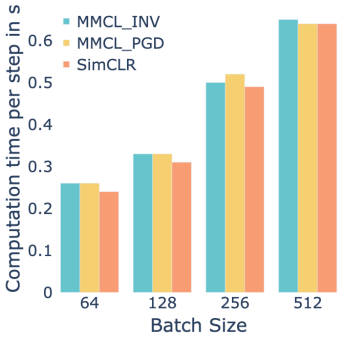

Figure 4: Computations (ImageNet).

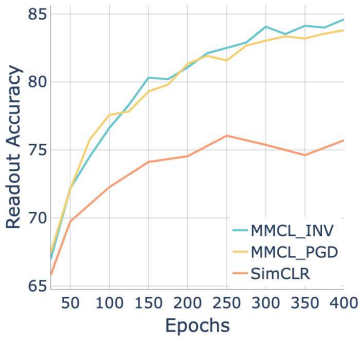

Figure 5: Convergence (STL-10).

Loss Variant	Modality	Negatives	top-1	R@1
MoCo-v2	RGB	2048	46.8	33.1
Ours	RGB	254	**52.45**	**45.6**
MoCo-v2	Flow	2048	66.8	45.2
Ours	Flow	254	**68.01**	**50.94**

Table 5: Video self-supervised learning on UCF-101 dataset.

Kernel ($K(x, y)$)	CIFAR100	STL10
Linear: $x^T y$	41.43%	74.82%
Tanh: $\tanh(-\gamma x^T y + \eta)$	54.53%	80.5%
RBF: $\exp(-\frac{\|x-y\|^2}{2\sigma^2})$	**55.35** %	**81.33**%

Table 6: Effect of kernel choice.

STL-10	64	128	256
SimCLR (Chen et al. 2020a)	74.21	77.18	80.15
DCL (Chuang et al. 2020)	76.72	81.09	84.26
HCL (Robinson et al. 2021)	80.39	83.98	87.44
Ours	80.11	**86.8**	**88.3**

Table 7: Accuracy (in %) against the batch size on STL-10.

hard negatives via support vectors is beneficial.

Computational time against batch size: In Figure 4, we show the time taken per iteration of MMCL variants against those of prior methods, such as SimCLR, for an increasing batch size. The computational cost of our inner optimization for finding the support vectors is directly related to the batch size. These experiments are done on ImageNet-1K with each RTX3090 GPU holding 64 images. We see that the time taken by MMCL is comparable to SimCLR.

Performances between MMCL Variants: In Table 8, we compare performances between MMCL variants: PGD and INV. We see that both variants outperform the SimCLR, while the two MMCL variants show similar performance. In Figure 5, we plot the convergence curves (readout accuracy) against training epochs. The plots clearly show that our variants converge to superior performances rapidly than the baseline.

Visualization of Support Vectors: Next, we qualitatively analyze if the support vectors found by MMCL are semantically meaningful. To this end, we use an MMCL model pre-trained on STL-10 dataset. We use a batch of examples as input to the model, and choose one of the examples from the batch as a positive and the remaining as negative. We then solve the MMCL objective to find α, where $\alpha = 0$ corresponds to non-support vectors, $\alpha = C$ are the misclassified points, and $\alpha \in [0, C)$ are the support vectors. In Figure 6, we show the positive point, and a set of samples from the batch and the respective α values. The figure clearly shows that object instances from a similar class gets a high α, suggesting that they lie on or inside the margin and contribute to the loss while batch samples that are irrelevant or easy negatives are not support vectors and do not contribute to the loss. For example, in Figure 6, the *yellow bird* is an easy negative for a *white truck* query image and our approach does not include that *bird* in the support set.

Longer Training: Our focus in the above experiments has

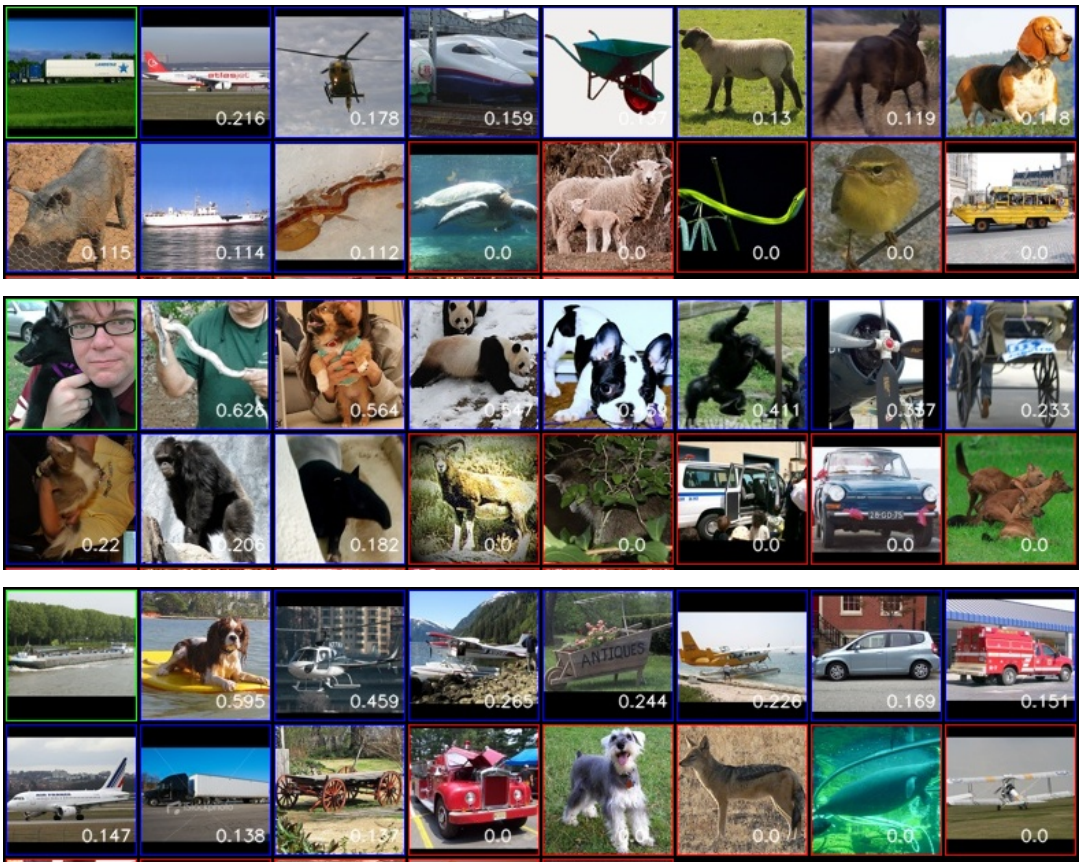

Figure 6: Visualizing Support Vectors : We visualize a query image (green box), corresponding support vectors (blue boxes) and non-support vectors (red boxes). We see that the support vectors are plausible hard negatives while in most cases the non-support vectors are easy negatives. The α corresponding to the various negatives is shown at the bottom left of each image.

Variant	CIFAR-100	STL-10
SimCLR	66	80.15
MMCL_{PGD}	**68.0**	88.03
MMCL_{INV}	**68.81**	**88.3**

Table 8: Performances on MMCL variants.

been in improving convergence and negative utilization with limited training. However, we see competitive performances on longer training as well. Using a batch size of 256 (510 negatives), our model reaches 66.5% in 200 and 69.9% in 400 epochs, compared to 62% and 64.5% respectively with same number of negatives in SimCLR(Chen et al. 2020a). Remarkably, our 100 epochs pre-trained models transfer better than PCL-v2's (Li et al. 2020b) 200 epoch models on most transfer learning tasks (see Table 1).

Conclusions

In this paper, we proposed a new contrastive learning framework, dubbed Max-Margin Contrastive Learning, using which we learn powerful deep representations for self-supervised learning by maximizing the decision margin separating data pseudo-labeled as positives and negatives. Our approach draws motivations from the classical support vector machines via modeling the selection of useful negatives through support vectors. We obtain consistent improvements over baselines on a variety of downstream tasks.

Acknowledgements

AS thanks Ketul Shah, Aniket Roy, Shlok Mishra, and Susmija Reddy for their feedback. AS and RC are supported by an ONR MURI grant N00014-20-1-2787. SS acknowledges support from NSF-TRIPODS+X:RES (1839258) and from NSF-BIGDATA (1741341).

Appendix

Additional Transfer Learning Experiments

In this section, we report results from several additional experiments we conducted analyzing the generalizability of MMCL-learned representations. Specifically, we used SUN397 (Xiao et al. 2010), Oxford-IIIT Pets (Parkhi et al. 2012), and VOC2007 (Everingham et al. 2010) for many-shot classification, ISIC (Tschandl, Rosendahl, and Kittler 2018;

Codella et al. 2019) and ChestX (Wang et al. 2017) for few-shot classification, and VOC 2007 (Everingham et al. 2010) for object detection. We follow the standard benchmarking protocol in (Ericsson, Gouk, and Hospedales 2021) for evaluating all our results. We report the top-1 accuracy metric on Food, CIFAR-10, CIFAR-100, SUN397, Stanford Cars, and DTD datasets. The mAP accuracy is reported for Aircraft, Pets, Caltech101, and Flowers and the 11-point mAP metric is reported on Pascal VOC 2007. For few-shot transfer experiments, we report the average accuracy along with a 95% confidence interval. We report AP for object detection on VOC and median angular error for surface normal estimation on NYUv2. Please refer to (Ericsson, Gouk, and Hospedales 2021) for additional details. For a fair comparison, we only compare to models which were pre-trained for a comparable number of epochs (upto 200) and batch-size of 256. To enable comparison with approaches like MoCo-v2 and MOCHI, we download their official pre-trained models (for a batch size of 256) and follow the same benchmarking process to report the numbers.[4] All results are reported in 9. We see that our approach consistently outperforms the prior approaches on most tasks and datasets which show the high-quality nature of representations learned using MMCL.

Additional Ablation Studies

In this section, we include some additional analyses and experiments to analyse the MMCL loss.

Effect of Slack Penalty C: In the main paper, we explored the effects of the slack penalty C on performance. In this section, we augment that study with an analysis of the behavior of our model towards a changing C. Specifically, note that when starting to train the model, the neural network weights are randomly initialized, and as a result, the generated contrastive learning features might be arbitrary that asking for a misclassification penalty might be unlikely to be useful. To empirically understand this conjencture, we train a model from scratch for 25 epochs using a slack $C = \infty$ (i.e., no misclassification is allowed) and then reduce C to a fixed value for the next 75 epochs. In Table 10, we provide the result of this analysis on CIFAR-100 and STL-10 datasets, where the legend shows the fixed value of C used. The table shows that there is some benefit of using this approach, for example, on STL-10 dataset, where we see that for C=10 shows a slight benefit over $C = \infty$ (i.e., not using the proposed approach). However, overall it looks like such a scheme does not reduce the performance dramatically, unless we underpenalize the misclassifications (such as using C=0.1, for example).

False Negative Correction: In the main paper, we argue that the use of slack allows to limit the effect that false negatives have on training by limiting the maximum contribution that they can make to the loss. An alternative idea would be to nullify all points that lie *inside* the margin (i.e., points that are potentially close to positives, and thus are perhaps false negatives). This would amount to the operation $\alpha[\alpha$ == C] = 0 after finding the optimal dual values α. For this experiment, we use a single C for the entire training (unlike the above experiment). The result of this experiment is shown in Figure 11. The table shows that FN correction has a significant impact on performance when using a small C, while insignificant for larger C. This is perhaps not unexpected given that using a small C will lead to a large number of misclassified points, which if removed from training, will lead to suboptimal contrastive learning, especially in the initial epochs of training. However, with larger values of C, it appears from the table that the impact of removing some points does not impact the performance much. This is perhaps because using the same value of C for all the training epochs is perhaps sub-optimal, and would need a schedule for changing this parameter. We plan to explore this idea in the future.

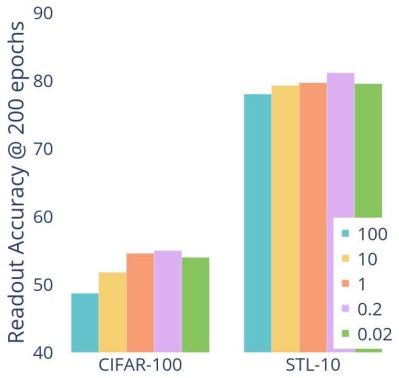

Figure 7: Effect of kernel bandwidth $1/\sigma^2$.

Influence of RBF Kernel Bandwidth σ: In this experiment we vary the RBF Kernel bandwidth for CIFAR-100 and STL-10. From Figure 7, we see that the performance can be influenced by the choice of σ, however, using $\sigma = 1$ performs reasonably well for these datasets.

Effect of Batch Size: In the main paper, we showed the effect of batch size on STL-10 (Table 7). In Table 12, we augment that with a similar study for CIFAR-100. Our performances on CIFAR-100 are similar, albeit the improvements are not as pronounced as in STL-10, perhaps because its a smaller dataset. Further, the trend in the tables show that similar to other methods, the performance of MMCL increases consistently with increasing batch sizes, while showing signs of diminishing returns as the batch sizes grows larger (e.g., grows from 256 to 512). We still consistently outperform the SimCLR baseline.

Convergence In Figure 5 of the main paper, we show the convergence analysis of training on STL-10. In Figure 8 we add a similar analysis for STL-10. Similar to results on STL-10, we see that the MMCL variants converge to superior performances more rapidly than the baseline.

Kernels with SimCLR: The loss used in vanilla SimCLR (and subsequently other contrastive learning approaches) is

[4]Note that comparing with the large batch size (as high as 4096) models and longer pre-trained models (as high as 1000 epochs) is not fair since they require as many as 16 high-performance GPUs (Caron et al. 2020) and very long pretraining times. For comparison, on our compute setup, training with a 256 batch size model for 100 epochs takes upwards of 3.5 days.

Method	Many-Shot classification Pets	SUN397	VOC2007	Few-Shot classification ISIC	ChestX	Object Det. VOC (AP)
Supervised	92.42	63.56	84.76	48.79 ± 0.53	29.26 ± 0.44	53.26
InsDis (Wu et al. 2018)	76.22	51.84	71.90	52.19 ± 0.53	29.13 ± 0.44	48.82
MoCo (He et al. 2020)	76.96	53.35	74.61	**53.79 ± 0.54**	**30.0 ± 0.43**	50.51
PCL-v1 (Li et al. 2020a)	86.98	58.40	82.08	38.01 ± 0.44	25.54 ± 0.43	**53.93**
PCL-v2 (Li et al. 2020a)	85.39	58.82	82.20	44.4 ± 0.52	28.28 ± 0.42	**53.92**
MoCo-v2 (Chen et al. 2020c)†	87.32	56.22	79.34	49.70 ± 0.51	29.48 ± 0.45	50.67
MoCHI (Kalantidis et al. 2020)†	87.4	57.74	79.73	49.0 ± 0.53	28.4 ± 0.4	52.61
Ours	**87.81**	**62.78**	**82.83**	48.79 ± 0.53	29.26 ± 0.44	50.73

Table 9: Transfer learning results. We transfer a ImageNet-pretrained model (using MMCL) on a range of downstream tasks and datasets. We compare with models pretrained using a similar batch size and epochs. Results on competing approaches are taken from (Ericsson, Gouk, and Hospedales 2021). †Models evaluated using publicly available checkpoints.

	CIFAR-100	STL-10
C=0.1	44.41	73.82
C=1	53.08	78.86
C=10	55.1	81.25
C=50	54.95	79.63
C=∞	55.1	79.42

Table 10: Reducing C over epochs.

	Readout Acc.
C=1	61.12
C=2	69
C=10	80.05
C=50	79.91
No Correction	79.63

Table 11: Using false negative correction for STL-10.

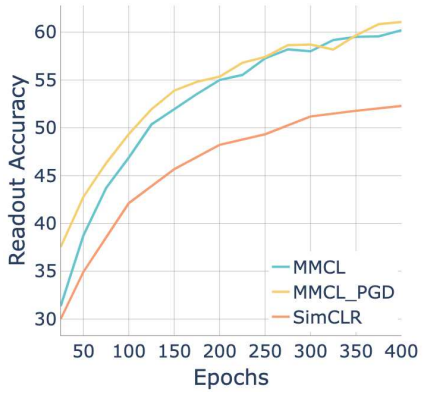

Figure 8: Convergence (CIFAR-100).

CIFAR-100	64	128	256	512
SimCLR	60.8	65	66	66.32
DCL	63.2	66.2	67.5	67.71
HCL	66.46	68.37	**69**	**70.31**
MMCL	65.82	**68.75**	**68.81**	**70.7**

Table 12: Accuracy (in %) against the batch size on CIFAR-100.

technically equivalent to using an RBF kernel, and one could perhaps choose other similarity kernels, e.g., tanh. To quantitatively evaluate this intuition and the impact of such a choice, we used a tanh kernel in SimCLR, and experimented with 5 different bandwidths. The results (Table 13) suggest that while there can be some improvements to SimCLR via selecting other kernel similarities, our proposed formulation demonstrates significantly better performance. This is because our proposed kernel SVM formulation produces a classification margin in an RKHS where the support set is automatically sparse and weighted, which to the best of our knowledge is difficult to be argued for in the SimCLR setup.

Using an Exact SVM Solver: Our initial approach was to use an exact SVM solver (using Qpth (Amos and Kolter 2017)) to estimate the max-margin hyperplane, however we found that our iterations were considerably slower (0.52 vs 3.07 for batch size 256).

Intuition for Slack Penalty C: Note that in contrast to standard SVMs where the penalty C is a trade off between the misclassification error and the margin, in our setup it has an even bigger role as the feature representation itself is evolving. Specifically, when using a larger C, the α weights on the misclassified points will be equal to C, and thus given our contrastive objective is using representations of data points linearly weighted by α, the backbone neural network will be updated via gradients to minimize this loss – thereby pushing these misclassified points out of the margin. Thus, in subsequent iterations, the focus of learning will be to increase the margin, as the number of misclassified points will be

Method	Readout @ 200
SimCLR	48.2
SimCLR tanh $\gamma = 1$	50.5
SimCLR tanh $\gamma = 0.5$	49.1
SimCLR tanh $\gamma = 0.1$	49.9
SimCLR tanh $\gamma = 2$	50
SimCLR tanh $\gamma = 10$	49.9
MMCL	**55**

Table 13: Use of Kernels with SimCLR. We see that while use of kernels does lead to an improvement for the SimCLR loss, our proposed formulation demonstrates significantly better performance

smaller. However, C must also account for an estimate of false negatives, and thus a larger value of C is perhaps not appropriate consistently. We found that an empirical evaluation of collisions is quite difficult, especially given that the network weights evolve over the epochs that collisions vanish as the contrastive loss drops. In Table 11, we attempted to define some heuristics to explicitly correct for false negatives (FN); our results show that when the penalty C is small, correcting for the FNs do show minor benefits (e.g., C=50). However, for larger C, the network weights are perhaps updated very quickly to fix the FN misclassification error in the early epochs that FNs are absent in subsequent epochs.

Other Details

PGD Solver: We used nesterov accelerated PGD solver to get faster convergence. The value of α before each pre-training step is randomly initialized before the PGD iterations. We use a step size of 0.001 and optionally a step size of $\frac{1}{\|\Delta\|_2}$.

ImageNet Experiments: We use an initial learning rate of 1.2 and following SimCLR we use a warmup and an annealing scheme. Our best model uses σ^2 of 5. To keep hyperparameter search tractable, we obtain top-1 validation accuracy after pre-training for 10 epochs and finetuning for 1 epoch. Based on exeriments with ImageNet-100, we found the PGD variant to work slightly better than the INV variant (80.7% vs 80.5%) and so we use PGD variant for all ImageNet experiments.

CIFAR-100 and STL-10 Experiments: We choose hyperparameters using the validation accuracy at 200 epochs based protocol mentioned in the main paper (Readout accuracy).

UCF-101 Experiments : We use a publicly available implementation for MoCo-v2 for video self supervised learning (Han, Xie, and Zisserman 2020). All the hyperparameters are the same except σ^2 of 0.5, $C = 1$, and learning rate of 1e-4. We use the PGD variant for experiments on UCF-101.

Limitations and Failure Cases

The proposed approach has some minor overheads. While, we did not see any significant difference in training times (as shown in Figure 4), they can be expensive atleast in the initial epochs of training. This is, for example, because, when the model is randomly initialized, the features produced are arbitrary and thus the PGD iterations in MMCL would need longer cycles to finish. However, we see that this trend is only for the initial few epochs and training speed improves over time. We could also perhaps use a different and better solver for box-constrained optimization that leads to much faster convergence (for example, (Kim, Sra, and Dhillon 2012) or OSQP (Schubiger, Banjac, and Lygeros 2020))

Potential Negative Impact

Our paper makes a fundamental contribution to the field of self-supervised and contrastive learning. We do not perceive any obvious negative social impacts of this work. In fact, self-supervision naturally helps to avoid certain biases that can emerge from labeling biases. That said, training self-supervised learning methods on already biased datasets could be harmful and potentially lead to the biases being inherited by the models during the fine-tuning stage.

References

Amos, B.; and Kolter, J. Z. 2017. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of International Conference on Machine Learning*, 136–145.

Arora, S.; Khandeparkar, H.; Khodak, M.; Plevrakis, O.; and Saunshi, N. 2019. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. In *Proceedings of International conference on Machine Learning*.

Becker, S.; and Hinton, G. E. 1992. Self-Organizing Neural Betwork that Discovers Surfaces in Random-Dot Stereograms. *Nature*, 355(6356): 161–163.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep Clustering for Unsupervised Learning of Visual Features. In *Proceedings of the European Conference on Computer Vision*, 132–149.

Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; and Joulin, A. 2020. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Proceedings of Advances in Neural Information Processing Systems*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A Simple Framework for Contrastive Learning of Visual Representations. In *Prroceedings of International conference on Machine Learning*, 1597–1607.

Chen, T.; Kornblith, S.; Swersky, K.; Norouzi, M.; and Hinton, G. E. 2020b. Big Self-Supervised Models are Strong Semi-Supervised Learners. *Proceedings of Advances in Neural Information Processing Systems*, 33: 22243–22255.

Chen, X.; Fan, H.; Girshick, R.; and He, K. 2020c. Improved Baselines with Momentum Contrastive Learning. *arXiv preprint arXiv:2003.04297*.

Chuang, C.-Y.; Robinson, J.; Lin, Y.-C.; Torralba, A.; and Jegelka, S. 2020. Debiased Contrastive Learning. In *Proceedings of Advanced in Neural Information Processing Systems*.

Coates, A.; Ng, A.; and Lee, H. 2011. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 215–223.

Codella, N.; Rotemberg, V.; Tschandl, P.; Celebi, M. E.; Dusza, S.; Gutman, D.; Helba, B.; Kalloo, A.; Liopyris, K.; Marchetti, M.; et al. 2019. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*.

Cortes, C.; and Vapnik, V. 1995. Support-Vector Networks. *Machine learning*, 20(3): 273–297.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised Visual Representation Learning by Context Prediction. In *Proceedings of the IEEE international conference on computer vision*, 1422–1430.

Ericsson, L.; Gouk, H.; and Hospedales, T. M. 2021. How Well Do Self-Supervised Models Transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5414–5423.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338.

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; and Ramanan, D. 2009. Object Detection with Discriminatively Trained Part-Based Models. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 32(9): 1627–1645.

Gidaris, S.; Singh, P.; and Komodakis, N. 2018. Unsupervised Representation Learning by Predicting Image Rotations. In *International Conference on Learning Representations*.

Gould, S.; Fernando, B.; Cherian, A.; Anderson, P.; Cruz, R. S.; and Guo, E. 2016. On Differentiating Parameterized Argmin and Argmax Problems with Application to Bi-level Optimization. *arXiv preprint arXiv:1607.05447*.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. A.; Guo, Z. D.; Azar, M. G.; et al. 2020. Bootstrap Your Own Latent: A New Approach To Self-Supervised Learning. *arXiv preprint arXiv:2006.07733*.

Guo, Y.; Codella, N. C.; Karlinsky, L.; Codella, J. V.; Smith, J. R.; Saenko, K.; Rosing, T.; and Feris, R. 2020. A Broader Study of Cross-Domain Few-Shot Learning. In *Proceedings of the European Conference on Computer Vision*, 124–141.

Gutmann, M.; and Hyvärinen, A. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 297–304.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1735–1742.

Han, T.; Xie, W.; and Zisserman, A. 2020. Self-supervised Co-Training for Video Representation Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, 5679–5690.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9729–9738.

Helber, P.; Bischke, B.; Dengel, A.; and Borth, D. 2019. EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7): 2217–2226.

Huynh, T.; Kornblith, S.; Walter, M. R.; Maire, M.; and Khademi, M. 2022. Boosting Contrastive Self-Supervised Learning with False Negative Cancellation. *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*.

Iscen, A.; Tolias, G.; Avrithis, Y.; and Chum, O. 2018. Mining on Manifolds: Metric Learning Without Labels. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7642–7651.

Kalantidis, Y.; Sariyildiz, M. B.; Pion, N.; Weinzaepfel, P.; and Larlus, D. 2020. Hard Negative Mixing for Contrastive Learning. In *Neural Information Processing Systems*.

Kim, D.; Sra, S.; and Dhillon, I. S. 2012. A non-monotonic method for large-scale non-negative least squares. *Optimization Methods and Software (OMS)*.

Kriege, N. M.; and Mutzel, P. 2012. Subgraph Matching Kernels for Attributed Graphs. In *Proceedings of International Conference on Machine Learning*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning Multiple Layers of Features from Tiny Images.

Larsson, G.; Maire, M.; and Shakhnarovich, G. 2016. Learning Representations for Automatic Colorization. In *Proceedings of European Conference on Computer Vision*, 577–593.

Lee, H.-Y.; Huang, J.-B.; Singh, M.; and Yang, M.-H. 2017. Unsupervised Representation Learning by Sorting Sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, 667–676.

Li, J.; Zhou, P.; Xiong, C.; and Hoi, S. 2020a. Prototypical Contrastive Learning of Unsupervised Representations. In *Proceedings of International Conference on Learning Representations*.

Li, J.; Zhou, P.; Xiong, C.; Socher, R.; and Hoi, S. C. 2020b. Prototypical Contrastive Learning of Unsupervised Representations. *arXiv preprint arXiv:2005.04966*.

Logeswaran, L.; and Lee, H. 2018. An Efficient Framework for Learning Sentence Representations. In *International Conference on Learning Representations*.

Malisiewicz, T.; Gupta, A.; and Efros, A. A. 2011. Ensemble of Exemplar-SVMs for Object Detection and Beyond. In *Proceedings of International Conference on Computer Vision*, 89–96.

Misra, I.; and Maaten, L. v. d. 2020. Self-Supervised Learning of Pretext-Invariant Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6707–6717.

Mohanty, S. P.; Hughes, D. P.; and Salathé, M. 2016. Using Deep Learning for Image-based Plant Disease Detection. *Frontiers in plant science*, 1419.

Noroozi, M.; and Favaro, P. 2016. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *Proceedings of European Conference on Computer Vision*, 69–84.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*.

Parkhi, O. M.; Vedaldi, A.; Zisserman, A.; and Jawahar, C. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, 3498–3505. IEEE.

Reddi, S. J.; Kale, S.; Yu, F.; Holtmann-Rice, D.; Chen, J.; and Kumar, S. 2019. Stochastic Negative Mining for Learning with Large Output Spaces. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 1940–1949. PMLR.

Robinson, J. D.; Chuang, C.-Y.; Sra, S.; and Jegelka, S. 2021. Contrastive Learning with Hard Negative Samples. In *International Conference on Learning Representations*.

Saeed, A.; Grangier, D.; and Zeghidour, N. 2021. Contrastive Learning of General-Purpose Audio Representations. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 3875–3879.

Santa Cruz, R.; Fernando, B.; Cherian, A.; and Gould, S. 2018. Visual Permutation Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3100–3114.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815–823.

Schubiger, M.; Banjac, G.; and Lygeros, J. 2020. GPU acceleration of ADMM for large-scale quadratic programming. *Journal of Parallel and Distributed Computing*, 144: 55–67.

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *Proceedings of European Conference on Computer Vision*, 746–760.

Smola, A. J.; and Schölkopf, B. 1998. *Learning with Kernels*, volume 4. MIT Press.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. UCF101: A Dataset of 101 Human Actions Classes from Videos in The Wild. *arXiv preprint arXiv:1212.0402*.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *Proceedings of International Conference on Learning Representations*.

Tian, Y.; Krishnan, D.; and Isola, P. 2020. Contrastive Multiview Coding. In *Proceedings of European Conference on Computer Vision*, 776–794.

Tschandl, P.; Rosendahl, C.; and Kittler, H. 2018. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1): 1–9.

Wang, J.; Cherian, A.; Porikli, F.; and Gould, S. 2018. Video Representation Learning using Discriminative Pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1149–1158.

Wang, X.; Peng, Y.; Lu, L.; Lu, Z.; Bagheri, M.; and Summers, R. M. 2017. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2097–2106.

Wu, Z.; Xiong, Y.; Yu, S. X.; and Lin, D. 2018. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3733–3742.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, 3485–3492. IEEE.

Xiao, T.; Wang, X.; Efros, A. A.; and Darrell, T. 2020. What Should Not Be Contrastive in Contrastive Learning. In *Proceedings of International Conference on Learning Representations*.

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K. P. 2018. Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification. In *Proceedings of the European Conference on Computer Vision*.

Yanardag, P.; and Vishwanathan, S. 2015. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1365–1374.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph Contrastive Learning with Augmentations. *Advances in Neural Information Processing Systems*, 33.

You, Y.; Gitman, I.; and Ginsburg, B. 2018. Large Batch Training of Convolutional Networks. *Proceedings of International Conference on Learning Representations*.

Zhan, X.; Xie, J.; Liu, Z.; Ong, Y.-S.; and Loy, C. C. 2020. Online Deep Clustering for Unsupervised Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6688–6697.

Zhang, L.; Qi, G.-J.; Wang, L.; and Luo, J. 2019. AET vs. AED: Unsupervised Representation Learning by Auto-Encoding Transformations rather than Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2547–2555.

Zhang, R.; Isola, P.; and Efros, A. A. 2016. Colorful Image Colorization. In *European conference on computer vision*, 649–666. Springer.

Zhuang, C.; Zhai, A. L.; and Yamins, D. 2019. Local Aggregation for Unsupervised Learning of Visual Embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6002–6012.