Unsupervised learning, Semi-Supervised Learning

R. Caruana, 1997, "Multitask Learning"

Multi-task learning (MTL)

It is an inductive transfer mechanism whose principal goal is to improve generalization performance. The MTL improves generalization by leveraging the domain-specific information contained in the training signals of *related* tasks. In effect, the training signals for the extra tasks serve as an inductive bias.

The standard methodology in machine learning is to learn one task at a time.

If the tasks can share what they learn, the learner may find it is easier to learn them together than in isolation. Thus, if we simultaneously train a net to recognize object outlines, shapes, edges, regions, subregions, textures, reflections, highlights, shadows, text, orientation, size, distance, etc., it may learn better to recognize complex objects in the real world.

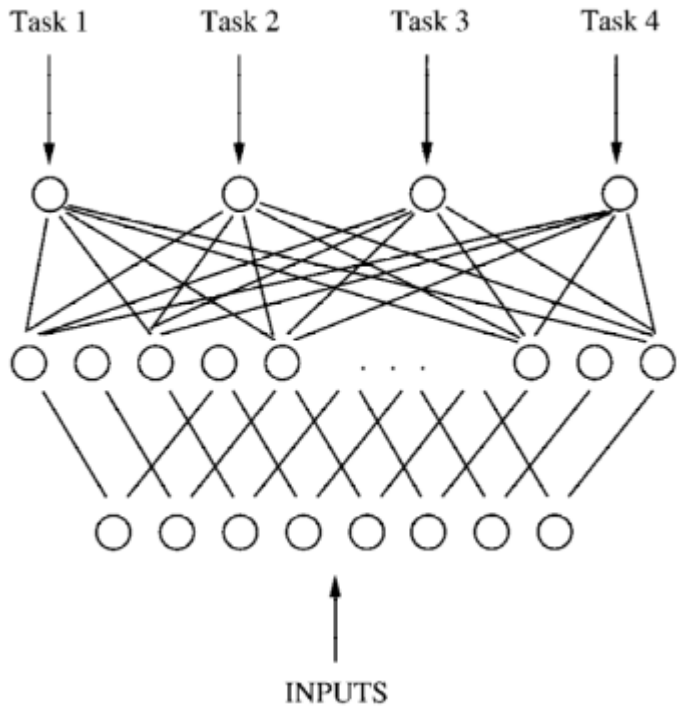


Fig. Multitask Backpropagation of four tasks with the same inputs

Z. Wu et al., 2018, "Unsupervised Feature Learning via Non-Parametric Instance Discrimination"

This paper attempts to learn good representations via unsupervised learning. It shows that a well-learned feature extractor is very useful to do other tasks such as object detection.

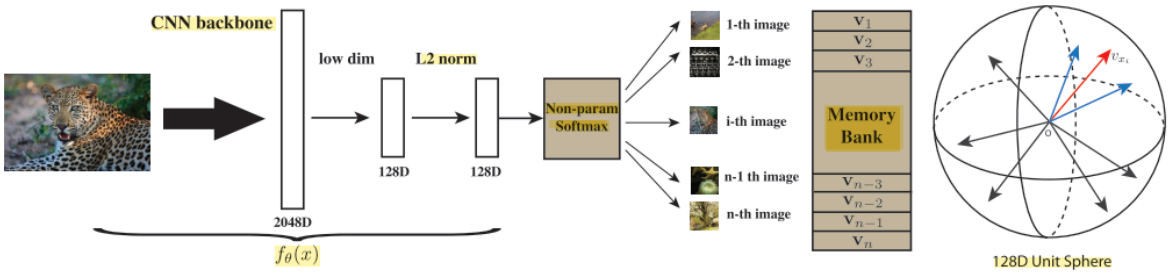


Figure 2: The pipeline of our unsupervised feature learning approach. We use a backbone CNN to encode each image as a feature vector, which is projected to a 128-dimensional space and L2 normalized. The optimal feature embedding is learned via instance-level discrimination, which tries to maximally scatter the features of training samples over the 128-dimensional unit sphere.

The goal is to learn an embedding function $v=f_\theta(x)$

without supervision. f_θ is a deep neural network with parameters θ, mapping image x to feature v. This embedding would induces a metric over the image space, as $d_\theta(x,y)=||f_\theta(x)-f_\theta(y)||$ for instances x and y

. A good embedding should map visually similar images closer to each other. This paper's novel unsupervised feature learning approach is *instance-level discrimination*. The authors treat *each image instance as a distinct class of its own* and train a classifier to distinguish between individual instance classes.

Non-Parametric Classifier

Under the conventional parametric softmax formulation, for image x

with feature $v=f_\theta(x)$, the probability of it being recognized as i

-th example is:

$$P(i|v)=\exp(w_i^T v) / \sum_{j=1}^{n}\exp(w_j^T v) \quad (1)$$

where w_j

is a weight vector for class j, and $w_j^T v$ measures how well v matches the j

-th class i.e., instance.

The authors propose a *non-parametric* variant of Eq.(1)

that replaces $w_j^T v$ with $v_j^T v$, and we enforce $||v||=1$ via a L2-normalization layer. This replacement allows instace-level comparison instead of comparing with a class prototype, w. Then, the probability $P(i|v)$

becomes:

$$P(i|v)=\exp(v_i^T v/\tau) / \sum_{j=1}^{n}\exp(v_i^T v/\tau) \quad (2)$$

where τ

is a temperature parameter that controls the concentration level of the distribution. τ is important for supervised feature learinng, and also necessary for tuning the concentration of v

on our unit sphere.

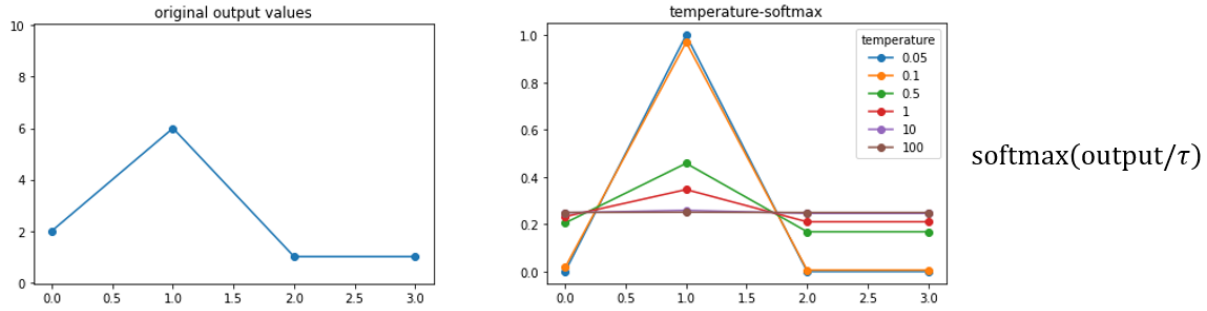


Fig. Effects of the temperature; The larger temperature ($\tau>1$) smoothes out the concentration, and the lower temperature ($\tau<1$) intensifies the concentration.

The learning objective is then to maximize the joint probability $\prod_{i=1}^{n} P_\theta(i|f_\theta(x_i))$

, or equivalently to minimize the negative log-likelihood (NLL) over the training set, as

$$J(\theta)=-\sum_{i=1}^{n}\log P(i|f_\theta(x_i))$$

Memory bank

To compute the probability $P(i|v)$

in Eq.(2), v_j for all the images are needed. Instead of exhaustively computing these representations every time, the authors maintain a feature memory bank V for storing them. Let $V=v_j$ be the memory bank and $f_i=f_\theta(x_i)$ be the feature of x_i. During each learning iteration, the representation f_i as well as the network parameters θ are optimized via SGD. Then, f_i is updated to V at the corresponding instance entry $f_i \rightarrow v_i$

.

Semi-supervised learning

A common scenario that can benefit from unsupervised learning is when we have a large amount of data of which only a small fraction are labeled. A natural semi-supervised learning approach is to first learn from the big unlabeled data and then fine-tune the model on the small labeled data. Namely, the representation learning is conducted with the big unlabeled dataset and the fine-tuning is conducted with the small labeled data.

Results

Image Classification Accuracy on ImageNet							
method	conv1	conv2	conv3	conv4	conv5	kNN	#dim
Random	11.6	17.1	16.9	16.3	14.1	3.5	10K
Data-Init [16]	17.5	23.0	24.5	23.2	20.6	-	10K
Context [2]	16.2	23.3	30.2	31.7	29.6	-	10K
Adversarial [4]	17.7	24.5	31.0	29.9	28.0	-	10K
Color [47]	13.1	24.8	31.0	32.6	31.8	-	10K
Jigsaw [27]	19.2	30.1	34.7	33.9	28.3	-	10K
Count [28]	18.0	30.6	34.3	32.5	25.7	-	10K
SplitBrain [48]	17.7	29.3	35.4	35.2	32.8	11.8	10K
Exemplar[3]			31.5			-	4.5K
Ours Alexnet	16.8	26.5	31.8	34.1	**35.6**	31.3	128
Ours VGG16	16.5	21.4	27.6	35.1	**39.2**	33.9	128
Ours Resnet18	16.0	19.9	29.8	39.0	**44.5**	**41.0**	128
Ours Resnet50	15.3	18.8	24.9	40.6	**54.0**	**46.5**	128

Table 2: Top-1 classification accuracy on ImageNet.

Image Classification Accuracy on Places							
method	conv1	conv2	conv3	conv4	conv5	kNN	#dim
Random	15.7	20.3	19.8	19.1	17.5	3.9	10K
Data-Init [16]	21.4	26.2	27.1	26.1	24.0	-	10K
Context [2]	19.7	26.7	31.9	32.7	30.9	-	10K
Adversarial [4]	17.7	24.5	31.0	29.9	28.0	-	10K
Video [44]	20.1	28.5	29.9	29.7	27.9	-	10K
Color [47]	22.0	28.7	31.8	31.3	29.7	-	10K
Jigsaw [27]	23.0	32.1	35.5	34.8	31.3	-	10K
SplitBrain [48]	21.3	30.7	34.0	34.1	32.5	10.8	10K
Ours Alexnet	18.8	24.3	31.9	**34.5**	33.6	30.1	128
Ours VGG16	17.6	23.1	29.5	33.8	**36.3**	32.8	128
Ours Resnet18	17.8	23.0	30.1	37.0	**38.1**	**38.6**	128
Ours Resnet50	18.1	22.3	29.7	42.1	**45.5**	**41.6**	128

Table 3: Top-1 classification accuracy on Places, based directly on features learned on ImageNet, without any fine-tuning.

The methods from 'Data-Init' to 'SplitBrain' refer to the pretext tasks; The linear evaluation is used with features from 'conv1' to 'conv5'. The result shows that the 'latter layers' and a 'deeper backbone' capture better representations.

Figure 5: Retrieval results for example queries. The left column are queries from the validation set, while the right columns show the 10 closest instances from the training set. The upper half shows the best cases. The lower half shows the worst cases.

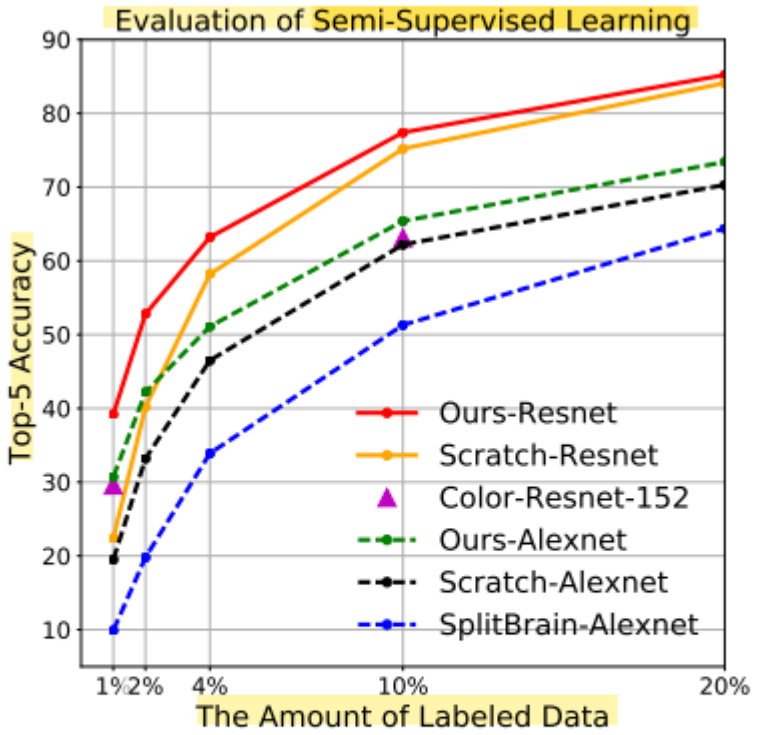


Figure 6: Semi-supervised learning results on ImageNet with an increasing fraction of labeled data (x axis). Ours are consistently and significantly better. Note that the results for colorization-based pretraining are from a deeper ResNet-152 network [19].

Method	mAP
AlexNet Labels†	56.8
Gaussian	43.4
Data-Init [16]	45.6
Context [2]	**51.1**
Adversarial [4]	46.9
Color [47]	46.9
Video [44]	47.4
Ours Alexnet	48.1

Method	mAP
VGG Labels†	67.3
Gaussian	39.7
Video [44]	60.2
Context [2]	61.5
Transitivity [45]	**63.2**
Ours VGG	60.5
ResNet Labels†	76.2
Ours ResNet	**65.4**

Table 6: Object detection performance on PASCAL VOC 2007 test, in terms of mean average precision (mAP), for supervised pretraining methods (marked by †), existing unsupervised methods, and our method.

A. Van Den Oord et al., 2019, "Representation Learning with Contrastive Predictive Coding" (CPC)

One of the most common strategies for unsupervised learning has been to predict future (forecasting), missing, or contextual information: "predictive coding"

In this paper, the authors propose a universal unsupervised learning approach to extract useful representations from high-dimensional data, which the authors call *Contrastive Predictive Coding*. The key insight of the proposed model is to learn such representations by *predicting the future in latent space* by using powerful autoregressive modes. This approach is able to learn useful representations achieving strong performance on four distinct domains: speech, images, text, and reinforcement learning.

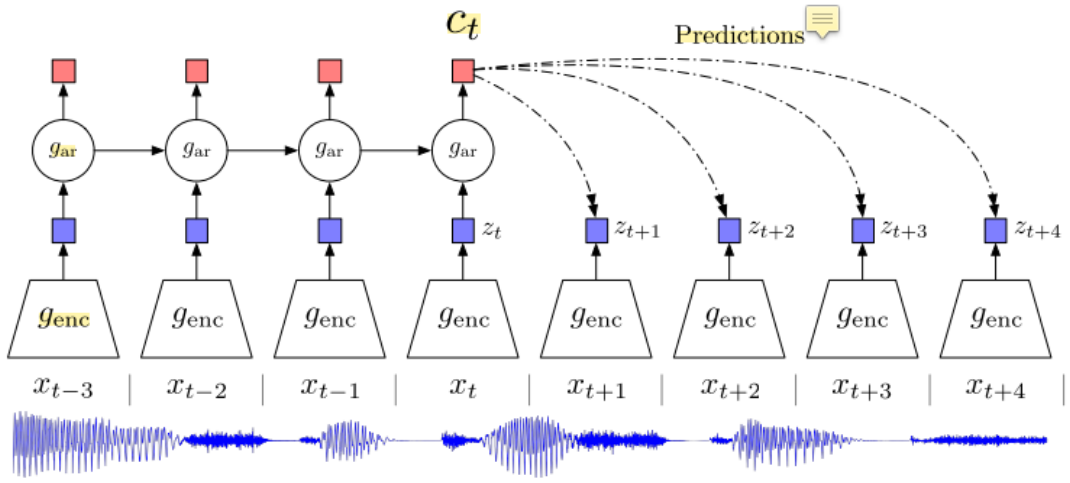


Figure 1: Overview of Contrastive Predictive Coding, the proposed representation learning approach. Although this figure shows audio as input, we use the same setup for images, text and reinforcement learning.

In the figure, g_{enc},g_{ar},z_t,c_t

denote an encoder, autoregressive model, latent representation, and context latent representation. Note that the prediction is made to predict the latent representation. The reason behind this is that we do not necessarily have to reconstruct the data in the high dimension (original input's dimension), which would be much harder. To do the predictive coding, prediction of the latent representation in the low dimension (latent embedding space) would be easier for the model.

Results

Method	ACC
#steps predicted	
2 steps	28.5
4 steps	57.6
8 steps	63.6
12 steps	64.6
16 steps	63.8
Negative samples from	
Mixed speaker	64.6
Same speaker	65.5
Mixed speaker (excl.)	57.3
Same speaker (excl.)	64.6
Current sequence only	65.2

Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

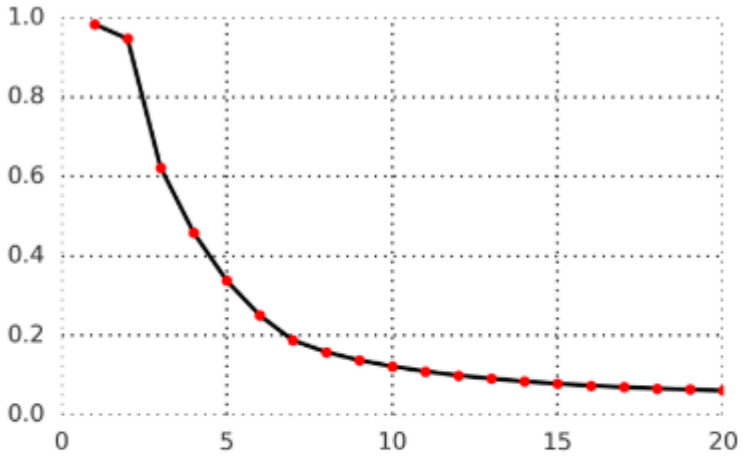


Figure 3: Average accuracy of predicting the positive sample in the contrastive loss for 1 to 20 latent steps in the future of a speech waveform. The model predicts up to 200ms in the future as every step consists of 10ms of audio.

It shows that prediction of the (kinda) far future is important for learning useful features which leads to the high classification performance although the prediction accuracy on the far future is not that high.

Long H. Nguyen et al., 2019, "Self-boosted Time-series Forecasting with multi-task and Multi-view Learning"

Traditional approaches usually require additional feature sets. However, adding more feature sets from different sources of data is not always feasible. In this paper, the authors propose a novel self-boosted mechanism in which the original univariate time series is decomposed into multivariate time series.

Multi-view

Having *multiple features* instead of one feature:

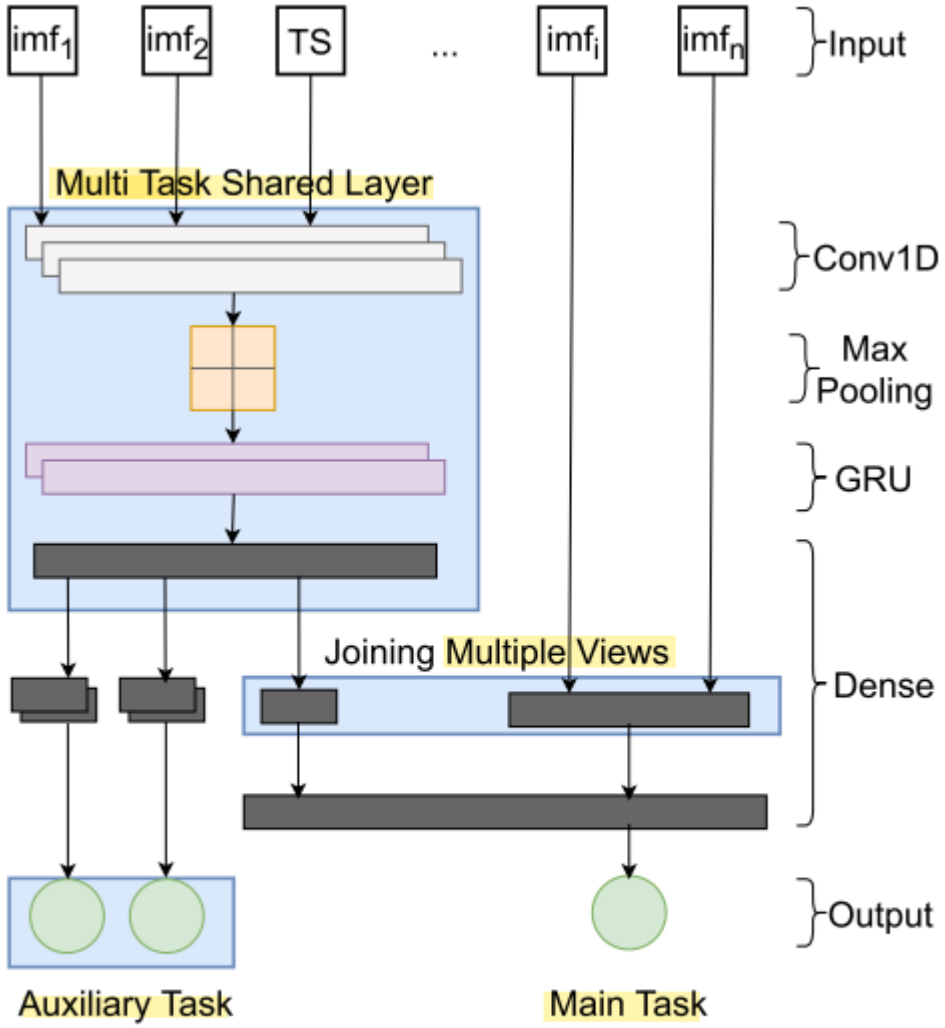


Figure 1: An overview of the multi-task and multi-view learning architecture in self-boosted time series forecasting framework.

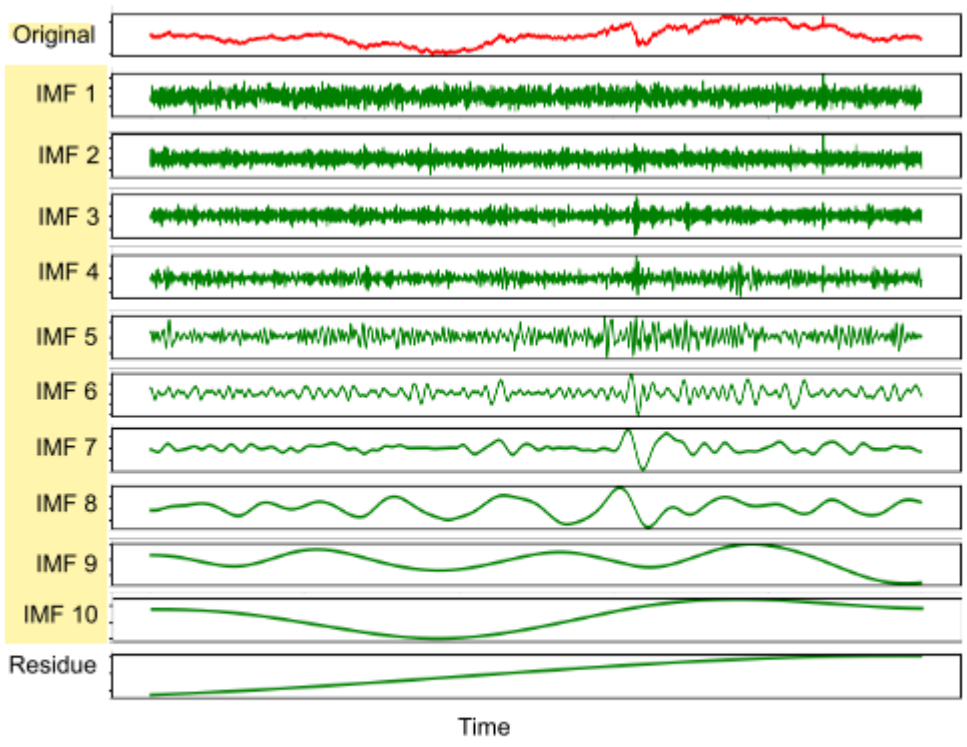


Figure 2: Exchange rate decomposed into intrinsic mode functions.

The decomposed closely related time series are fed to build a multi-task learning model while the decomposed loosely related time series group is utilized for multi-vew learning. The main task is forecasting at the forecast horizon, and the auxiliary tasks are forecasting related time series.

Empirical Ensemble Decomposition (EEDM) algorithm

It decomposes an original one-dimensional (univariate) time series into multiple components as shown in the above figure (Fig. 2). These decomposed time series can be categorized into two groups: 1) Closely related group to the original time series, 2) Loosely related group to the original time series. The similarity is measured by the correlation coefficient.

The EEMD decomposition transforms non-linear, non-stationary time series to stationary time series and can be useful for forecasting performance.

Intrinsic mode functions (IMF)

It is any time-varying function with the same number of extrema and zero crossings, whose envelopes are symmetric with respect to zero.

Results

Three datasets are used: 1) Electricity, 2) Exchange rate, 3) Air temperature. All models forecast a one-time step forward.

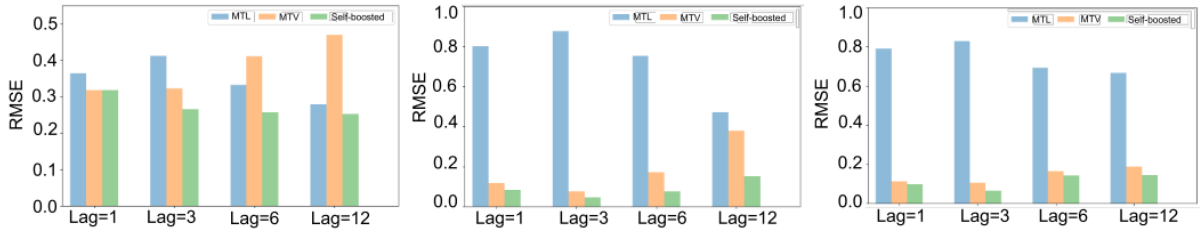


Figure 3: Performance comparison of multi-task learning (MTL), multi-view Learning (MTV) and the proposed method across all datasets (left: electricity consumption; middle: air temperature; right: exchange rate). X-axis is the lag time and Y-axis is the normalized RMSE value.

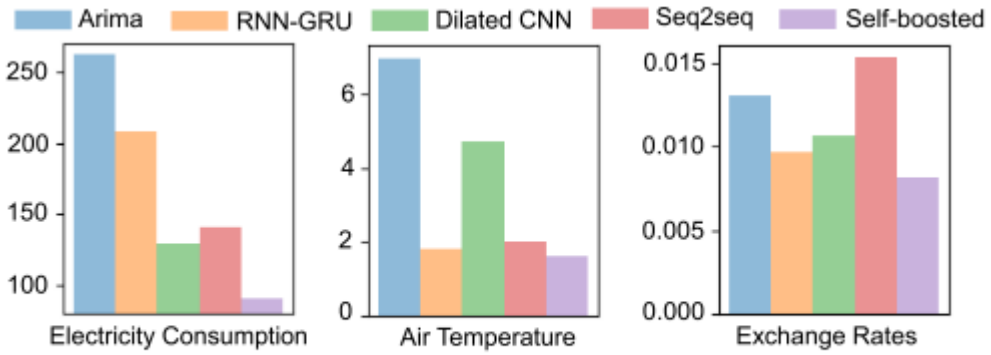


Figure 4: Average RMSE comparison on each dataset.

M. Caron et al., 2018, "Deep clustering for unsupervised learning of visual features"

"DeepCluster" is proposed. It iterates between clustering with k-means the features produced by the ConvNet and updating its weights by predicting the cluster assignments as pseudo-labels in a discriminative loss.

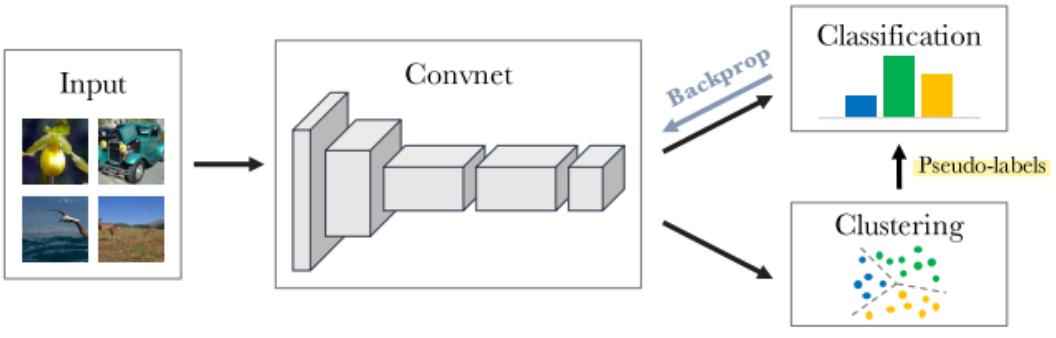


Fig. 1: Illustration of the proposed method: we iteratively cluster deep features and use the cluster assignments as pseudo-labels to learn the parameters of the convnet

For clustering, the k-means is used.

Related Work

A popular form of unsupervised learning, called "self-supervised learning" uses the pretext tasks (up until sometime in 2018). The pretext task-approaches are domain-dependent, while the proposed algorithm in this paper is not.

Weight Optimization

Weights of a feature extractor and a linear classifier are learned by optimizing the following equation:

$\min_{\theta,W} \frac{1}{N} \sum_{n=1}^{N} l(g_W(f_\theta(x_n)), y_n) \quad (1)$

where θ

, W are the weights of the feature extractor and linear classifier, x_n, y_n denote each image and corresponding label in $\{0,1\}^k$, respectively. This label represents the image's membership to one of k possible predefined classes. f, g denote the feature extractor and linear classifier, respectively. l

denotes the multinomial logistic loss, also known as the negative log-softmax function.

Unsupervised Learning by Clustering

The authors cluster the output of the convnet and use the subsequent cluster assignments as "pseudo-labels" to optimize Eq.(1)

. This deep clustering (DeepCluster) approach iteratively learns the features and groups them.

The k

-means takes a set of vectors as input, in our case the features $f_\theta(x_n)$ produced by the convnet, and clusters them into k distinct groups based on a geometric criterion. More precisely, it jointly learns a $d \times k$ centroid matrix C and the cluster assignments y_n of each image n

by solving the following problem:

$$\min_{C \in \mathbb{R}^{d \times k}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n \in \{0,1\}^k} \|f_\theta(x_n) - C y_n\|_2^2 \quad \text{such that} \quad y_n^\top 1_k = 1. \qquad (2)$$

Solving this problem provides a set of optimal assignments $(y^*_n)_{n<=N}$

and a centroid matrix C^*

. These assignments are then used as pseudo-labels.

Overall, DeepCluster alternates between clustering the features to produce pseudo-labels using Eq.(2)

and updating the parameters of the convnet by predicting these pseudo-labels using Eq.(1)

.

Avoiding Trivial Solutions

In this section, the authors briefly describe the causes of these trivial solutions and give simple and scalable workarounds.

Empty clusters

A discriminative model learns decision boundaries between classes. An optimal decision boundary is to assign all of the inputs to a single cluster (distance is minimized to zero by all the inputs being in the same class). This issue is caused by the absence of mechanisms to prevent the empty clusters. A common trick used in feature quantization consists in automatically reassigning empty clusters during the k

-means optimization. More precisely, when a cluster becomes empty, we randomly select a non-empty cluster and use its centroid with a small random perturbation as the new centroid for the empty cluster. We then reassign the points belonging to the non-empty cluster to the two resulting clusters.

Trivial parameterization

If the vast majority of images are assigned to a few clusters, the parameters θ

will exclusively discriminate between them. In the most dramatic scenario where all but one cluster are singleton, minimizing Eq.(1) leads to a trivial parameterization where the convnet will predict the same output regardless of the input. This issue also arises in supervised classification when the number of images per class is highly unbalanced. A strategy to circumvent this issue is to sample images based on a uniform distribution over the classes, or pseudo-labels. This is equivalent to weight the contribution of an input to the loss function in Eq.(1)

by the inverse of the size of its assigned cluster.

M. Caron et al., 2021, Unsupervised Learning of Visual Features by Contrasting Cluster Assignments (SwAV)

SwAV takes advantage of contrastive methods without requiring to compute pairwise comparisons. Specifically, this method simultaneously clusters the data while enforcing consistency between *cluster assignments* produced for different augmentations (or "views")

of the same image, instead of comparing features directly as in contrastive learning. Simply put, the authors use a "swapped" prediction mechanism where they predict the code (i.e., cluster assignments) of a view from the representation of another view.

Introduction

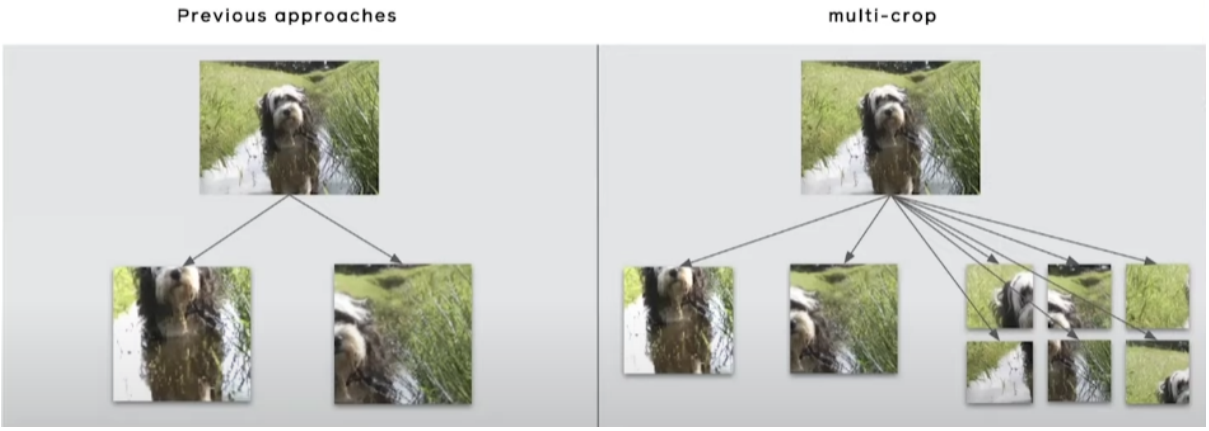
Many recent SOTA methods build upon the "instance discrimination task" that considers each image of the dataset (or "instance") and its transformations as a separate class (Z. Wu et al., 2018, "Unsupervised Feature Learning via Non-Parametric Instance Discrimination"). This task yields representations that are able to discriminate between different images, while achieving some invariance to image transformations. The recent self-supervised methods that use the instance discrimination rely on a combination of the following two elements: 1) a contrastive loss, 2) a set of image transformations.

The contrastive loss explicitly compares pairs of image representations to push away representations from different images while [pulling together those from transformations, or views, of the same image. Since computing all the pairwise comparisons on a large dataset is not practical, most implementations approximate the loss by reducing the number of comparisons to random subsets of images during training (Z. Wu et al., 2018, "Unsupervised Feature Learning via Non-Parametric Instance Discrimination").

An alternative to approximate the loss is to approximate the task - that is to relax the instance discrimination problem. For example, "clustering-based methods" discriminate between groups of images with similar features instead of individual images (M. Caron et al., 2018, "Deep Clustering for Unsupervised Learning of Visual Features"). The objective in clustering is tractable, but it does not scale well with the dataset as it requires a pass over the entire dataset to form image "codes" (i.e., cluster assignments) that are used as targets during training.

In this paper, the authors use a different paradigm and propose to *compute the codes online* while enforcing consistency between codes obtained from views of the same image. Comparing cluster assignments allows to contrast different image views while not relying on explicit pairwise feature comparisons. Specifically, the authors propose a simple *"swapped" prediction problem where they predict the code of a view from the representation of another view*. The representation features are learned by **Sw**apping **A**ssignments between multiple **V**iews of the same image (**SwAV**). *The feature and the codes are learned online* allowing this method to scale to potentially unlimited amounts of data. In addition, It does not need any large memory bank.

The authors also propose an image transformation, called "multi-crop". It uses smaller-sized images to increase the number of views while not increasing the memory or computational requirements during training. Directly working with downsized images introduces a bias of the features, which can be avoided by using a mix of different sizes.



Related Work

Instance and contrastive learning

Instance-level classification considers each image in a dataset as its own class. In contrast to this line of works, the authors in this paper avoid comparing every pair of images by mapping the image features to a set of trainable prototype vectors.

Clustering for deep representation learning

This paper's work is related to clustering-based methods. [M. Caron, 2018, DeepCluster] shows that *k*-means assignments can be used as pseudo-labels to learn visual representations. In this paper, the authors keep the soft assignment produced by the Sinkhorn-Knopp algorithm instead of approximating it into a hard assignment. Besides, unlike [M. Caron, 2018], the authors obtain online assignments which allows the proposed method to scale to any dataset size.

Method

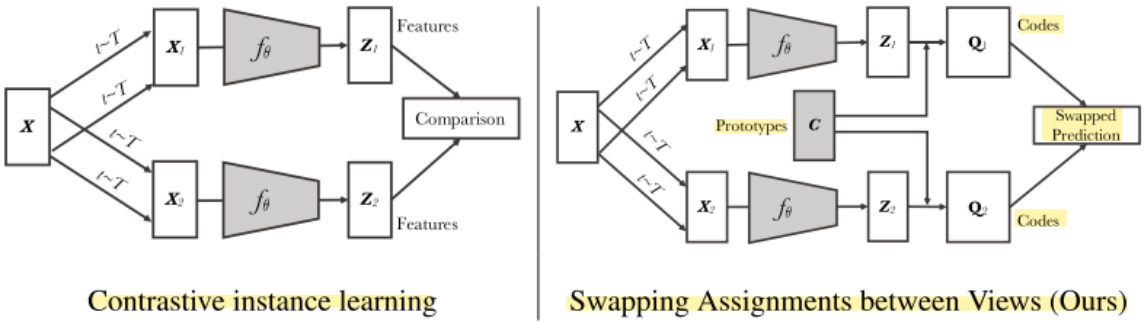


Figure 1: **Contrastive instance learning (left) *vs.* SwAV (right).** In contrastive learning methods applied to instance classification, the features from different transformations of the same images are compared directly to each other. In SwAV, we first obtain "codes" by assigning features to prototype vectors. We then solve a "swapped" prediction problem wherein the codes obtained from one data augmented view are predicted using the other view. Thus, SwAV does not directly compare image features. Prototype vectors are learned along with the ConvNet parameters by backpropragation.

Typical clustering-based methods are offline in the sense that they alternate between a cluster assignment step where image features of the entire dataset are clustered, and a training step where the cluster assignments, i.e., "codes" are predicted for different image views. In the proposed method, the authors do not consider the codes as a target, but only enforce consistent mapping between views of the same image. The method can be interpreted as a way of contrasting between multiple image views by comparing their cluster assignments instead of their features.

More precisely, we compute a code from an augmented version of the image and predict this code from other augmented versions of the same image. Given two image features z_t

and z_s from two different augmentations of the same image, we compute their codes q_t and q_s y matching these features to a set of K prototypes $\{c_1,\cdots,c_K\}$

. We then setup a "swapped" prediction problem with the following loss function:

$L(z_t,z_s)=l(z_t,q_s)+l(z_s,q_t)$

where the function \mathcal{l}(\boldsymbol{z}, \boldsymbol{q}) measures the fit between features z

and a code q. Intuitively, this method compares the features z_t and z_s using the intermediate codes q_t and q_s

. If these two features capture the same information, it should be possible to predict the code from the other feature.

Online clustering

Each image x_n

is transformed into an augmented view x_{nt} by applying a transformation t sampled from the set T of image transformations. The augmented view is mapped to a vector representation by applying a non-linear mapping f_θ to x_{nt}. The feature is then projected to the unit sphere, i.e., $z_{nt}=f_\theta(x_{nt})/||f_\theta(x_{nt})||_2$. We then compute a code q_{nt} from tihs feature by mapping z_{nt} to a set of K trainable prototypes vectors, $c_1,\cdots,c_K$. We denote by C the matrix whose columns are the $c_1,\cdots,c_k$

. We now describe how to compute these codes and update the prototypes online.

Swapped prediction problem

The loss function in Eq.(1)

has two terms that setup the "swapped" prediction problem of predicting the code q_t from the feature z_s, and q_s from z_t. Each term represents the cross-entropy loss between the code

and the probability obtained by taking a softmax of the dot products of z_i and all prototypes in C

, i.e.,

$$\ell(\mathbf{z}_t, \mathbf{q}_s) = -\sum_k \mathbf{q}_s^{(k)} \log \mathbf{p}_t^{(k)}, \quad \text{where} \quad \mathbf{p}_t^{(k)} = \frac{\exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_{k'}\right)}. \tag{2}$$

where τ

is a temperature parameter. It controls the concentration level of the distribution such that .It controls the concentration level of the distribution such that the larger temperature ($\tau>1$) smoothes out the concentration, and the lower temperature ($\tau<1$

) intensifies the concentration. Taking this loss over all the images and pairs of data augmentations leads to the following loss function for the swapped prediction problem:

$$-\frac{1}{N}\sum_{n=1}^{N}\sum_{s,t\sim\mathcal{T}}\left[\frac{1}{\tau}\mathbf{z}_{nt}^\top \mathbf{C}\mathbf{q}_{ns} + \frac{1}{\tau}\mathbf{z}_{ns}^\top \mathbf{C}\mathbf{q}_{nt} - \log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{nt}^\top \mathbf{c}_k}{\tau}\right) - \log\sum_{k=1}^{K}\exp\left(\frac{\mathbf{z}_{ns}^\top \mathbf{c}_k}{\tau}\right)\right].$$

This loss function is jointly minimized with respect to the prototypes C

and the parameters θ of the image encoder f_θ used to produce the features $(z_{nt})_{n,t}$

Multi-crop: Augmenting views with smaller images

We use two standard resolution crops and sample V

additional low-resolution crops that cover only small parts of the image. Using low-resolution images ensures only a small increase in the compute cost. Specifically, we generalize the loss of Eq.(1)

:

$$L(\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \ldots, \mathbf{z}_{t_{V+2}}) = \sum_{i\in\{1,2\}}\sum_{v=1}^{V+2}\mathbf{1}_{v\neq i}\ell(\mathbf{z}_{t_v}, \mathbf{q}_{t_i}). \tag{6}$$

Handcrafted pretext tasks

this video

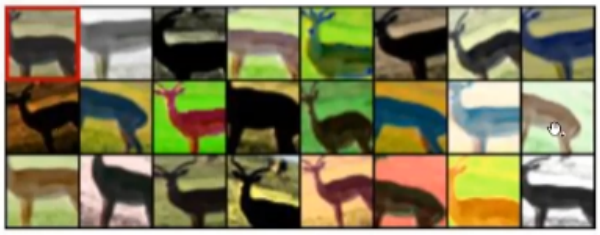
Exemplar, 2014

- "Discriminative unsupervised feature learning with exemplar convolutional neural networks", 2014 NIPS

Regions containing considerable gradients

  - Randomly sample $N \in [50, 32000]$ patches of size 32x32 from different images
  - Apply a various transformations to a randomly sampled "seed" image patch
  - Train to classify these exemplars as same class → **cannot be scalable to large datasets!**

Seed



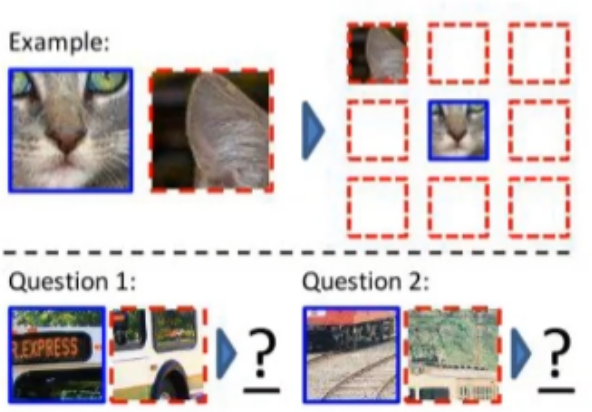
Train with STL-10 dataset (96x96)

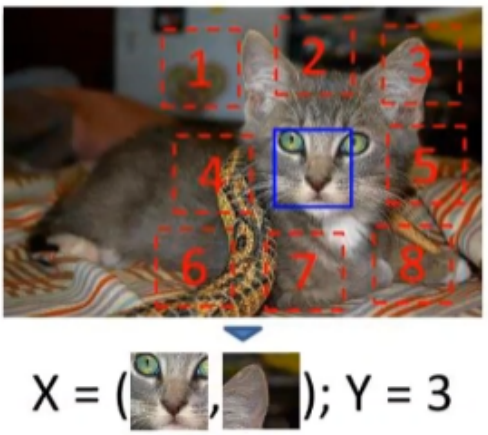
Transformations

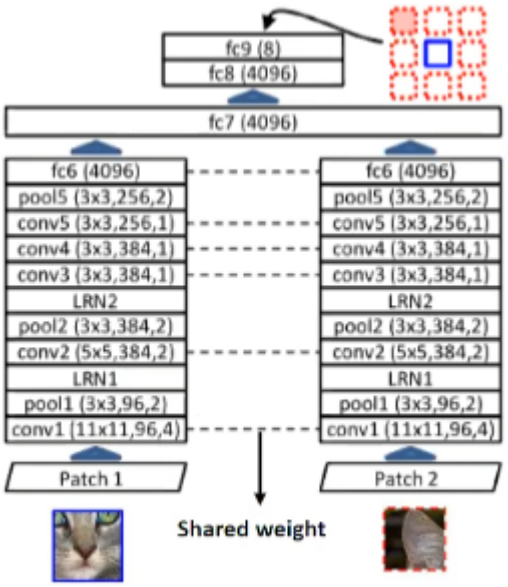
- translation: vertical or horizontal translation by a distance within 0.2 of the patch size;
- scaling: multiplication of the patch scale by a factor between 0.7 and 1.4;
- rotation: rotation of the image by an angle up to 20 degrees;
- contrast 1: multiply the projection of each patch pixel onto the principal components of the set of all pixels by a factor between 0.5 and 2 (factors are independent for each principal component and the same for all pixels within a patch);
- contrast 2: raise saturation and value (S and V components of the HSV color representation) of all pixels to a power between 0.25 and 4 (same for all pixels within a patch), multiply these values by a factor between 0.7 and 1.4, add to them a value between −0.1 and 0.1;

Relative Patch Location, 2015

- Aim to Self-Supervised Learning for image data using **context prediction**
- The algorithm must guess the position of one patch relative to the other



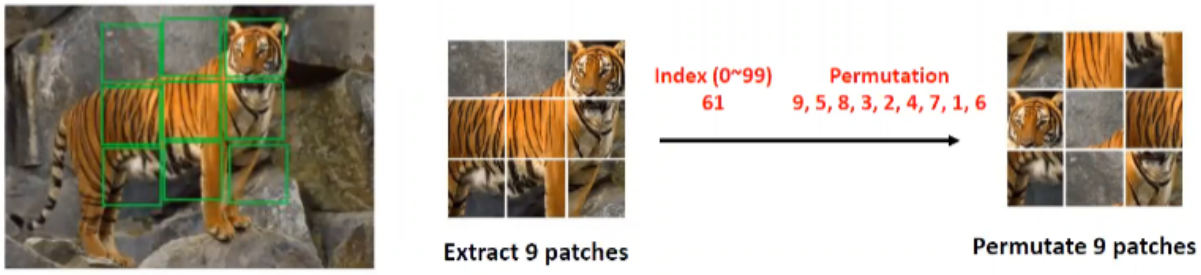




LRN: Local Response Normalization (= a kind of regularization layer)

Jigsaw Puzzles, 2016

- Recover relative spatial position of 9 randomly sampled image patches after random permutation
- 9! = 362,880 permutations, so remove similar permutations → use predefined permutation set (100)
- Network output is 100-d vector that predicts a permutation index



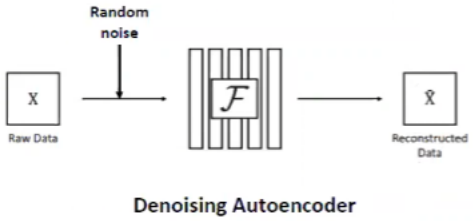
Sample image

Extract 9 patches

Permutate 9 patches

Autoencoder-Base Approaches

- Denoising Autoencoder, Context Autoencoder, Colorization, Split-brain Autoencoders
- Learn image features from reconstructing images without any annotation



Denoising Autoencoder

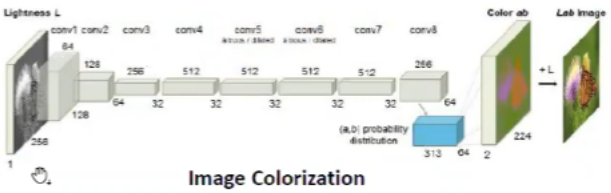
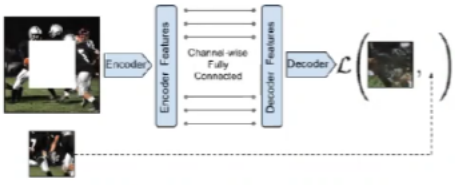
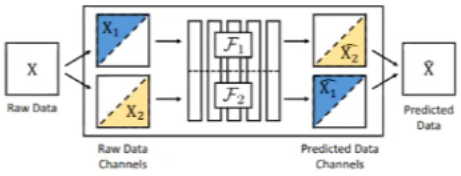


Image Colorization



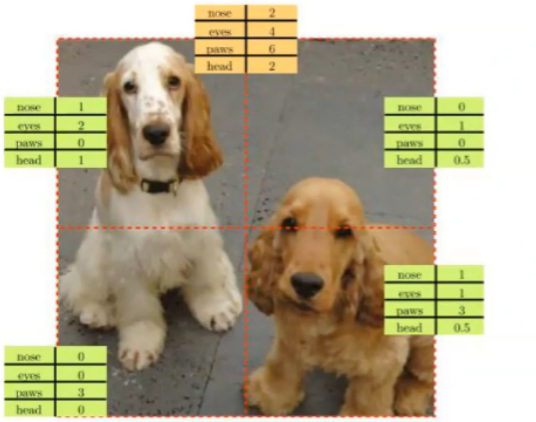
Context Autoencoder



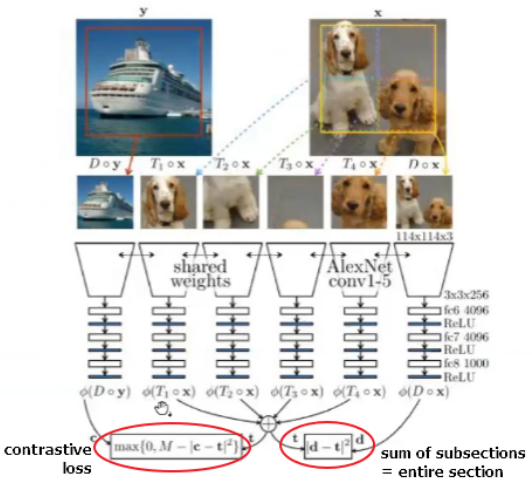
Split-Brain Autoencoder

Count, 2017

- **The number of visual primitives** in the whole image should match **the sum of those** in each tile
- Also, a feature that counts visual primitives should not be affected by scale, translation and rotation
- In this work, use downsampling(D) and tiling(T_j, j=1, 2, 3, 4)



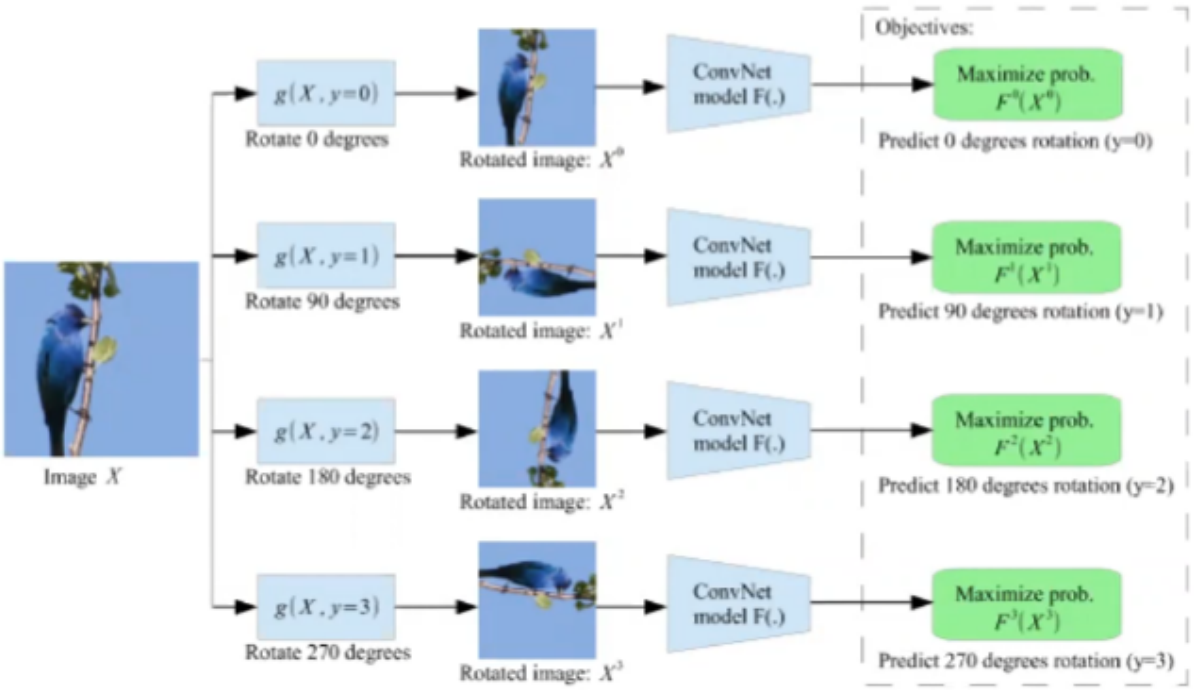
* This values are not label, just example for explanation!!!!



Multi-task, 2017

Combining several pretext tasks (= Relative path location + Colorization + Exemplar + Motion Segmentation)

Rotations, 2018



K. He et al., 2019, "Momentum contrast for unsupervised visual representation learning" (MoCo)

1. Introduction

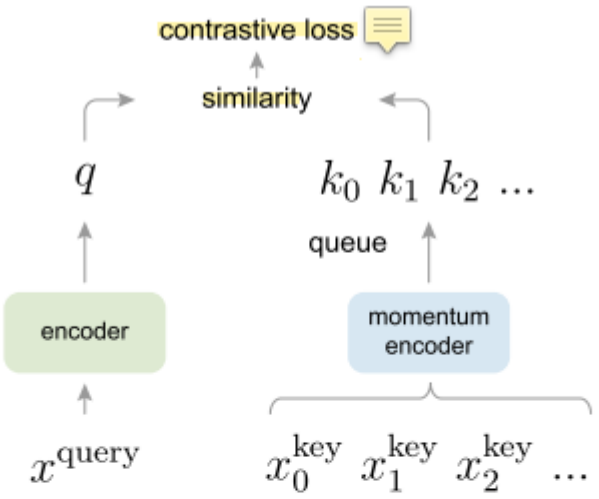


Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, ...\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder. This method enables a large and consistent dictionary for learning visual representations.

The "keys" in the dictionary are sampled from data (e.g., images or patches) and are represented by an encoder network. Unsupervised learning trains encoders to perform dictionary look-up: an encoded "query" should be similar to its matching key and dissimilar to others. Learning is formulated as minimizing a contrastive loss.

Momentum Contrast (MoCo) is a way of building large and consistent dictionaries for unsupervised learning with a contrastive loss. We maintain the dictionary as a *queue* of data samples: the encoded representations of the current mini-batch are enqueued, and the oldest are dequeued. The queue decouples the dictionary size from the mini-batch size, allowing it to be large. Moreover, as the dictionary keys come from the preceding several mini-batches, a *slowly progressing* key encoder, implemented as a momentum-based moving average of the query encoder, is proposed to maintain consistency.

MoCo is a mechanism for building dynamic dictionaries for contrastive learning, and can be used with various pretext tasks. In this paper, we follow a simple *instance discrimination*

task: a query matches a key if they are encoded views (e.g., different crops) of the same image.

3. Method

3.1. Contrastive Learning as Dictionary Look-up

Contrastive learning, and its recent developments, can be thought of as training an encoder for a *dictionary look-up* task, as described next.

Consider an encoded query q

and a set of encoded samples $\{k_0, k_1, k_2, \cdots\}$ that are the keys of a dictionary. Assume that there is a single key (denoted as k_+) in the dictionary that q matches. A contrastive loss is a function whose value is low when q is similar to its positive key k_+ and dissimilar to all other keys (considered negative keys for q

). With similarity measured by dot product, a form of a contrastive loss function, called InfoNCE (A. Oord et al., 2018, "Representation learning with contrastive predictive coding"), is considered in this paper:

$$L_q = -\log(\exp(q \cdot k_+/\tau)/(\sum_{i=0}^{K} \exp(q \cdot k_i/\tau)))$$

where τ

is a temperature hyper-parameter (In this paper, it's set to 0.07). It controls the concentration level of the distribution such that the larger temperature ($\tau > 1$) smoothes out the concentration, and the lower temperature ($\tau < 1$) intensifies the concentration. The sum is over one positive and K negative samples. Intuitively, this loss is the log loss of a $(K+1)$-way softmax-based classifier that tries to classify q as k_+

(i.e., log loss of softmax is equal to the [cross entropy loss implemented in PyTorch]).

The contrastive loss serves as an unsupervised objective function for training the encoder networks that represent the queries and keys. In general, the query representation is $q = f_q(x_q)$

where f_q is an encoder network and x_q is a query sample (likewise, $k = f_k(x_k)$). Their instantiations depend on the specific pretext task. The input x_q and x_k can be images, patches, or context consisting a set of patches. The network f_q and f_k

can be identical, partially shared, or different (In case of MoCo, they are identical with some time delay).

3.2. Momentum Contrast

Our hypothesis is that good features can be learned by a *large* dictionary that covers a rich set of negative samples, while the encoder for the dictionary keys is ket as consistent as

possible despite its evolution. Based on this motivation, the authors present Momentum Contrast.

Dictionary as a queue

At the core of our approach is maintaining the dictionary as a *queue* of data samples. This allows us to reuse the encoded keys from the immediate preceding mini-batches. The introduction of a queue decouples the dictionary size from the mini-batch size. Our dictionary size can be much larger than a typical mini-batch size, and can be flexibly and independently set as a hyper-parameter. The samples in the dictionary are progressively replaced. The current mini-batch is enqueued to the dictionary, and the oldest mini0batch in the queue is removed. The dictionary always represents a sampled subset of all data, while the extra computation of maintaining this dictionary is manageable. Moreoever, removing the oldest mini-batch can be beneficial, because its encoded kyes are the most outdated and thus the least consistent with the newest ones.

Momentum update

Formally, denoting the parameters of f_k

as θ_k and those of f_q as θ_q, we update θ_k

by:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \quad (2)$$

Here $m \in [0,1)$

is a momentum coefficient. Only the parameters θ_q are updated by back-propagation. The momentum update in $Eq.(2)$ makes θ_k evolve more smoothly than θ_q. As a result, though the kyes in the queue are encoded by different encoders (in different mini-batches), the difference among these encoders can be made small. In experiments, a relatively large momentum (e.g., $m=0.999$, our default) works much better than a smaller values (e.g., $m=0.9$

), suggesting that a slowly evolving key encoder is a core to making use of a queue.

Relation to previous mechanisms

MoCo is a general mechanism for using contrastive losses. We compare it with two existing general mechanisms in the following figure. We compare it with two existing general mechanisms. They exhibit different properties on the dictionary size and consistency:

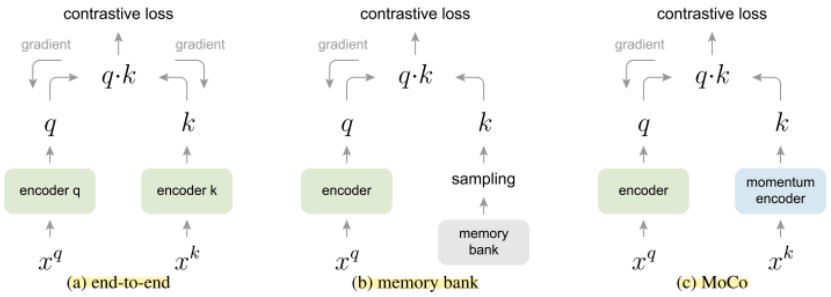


Figure 2. **Conceptual comparison of three contrastive loss mechanisms** (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. **(a)**: The encoders for computing the query and key representations are updated ***end-to-end*** by back-propagation (the two encoders can be different). **(b)**: The key representations are sampled from a ***memory bank*** [61]. **(c)**: ***MoCo*** encodes the new keys on-the-fly by a momentum-updated encoder, and maintains a queue (not illustrated in this figure) of keys.

The ***end-to-end*** update by back-propagation is a natural mechanism (Fig. 2a). It uses samples in the current mini-batch as the dictionary, so the keys are consistently encoded (by the same set of encoder parameters). But the dictionary size is coupled with the mini-batch size, limited by the GPU memory size.

Another mechanism is the ***memory bank*** approach proposed by [Z. Wu, 2018, "Unsupervised feature learning via non-parametric instance discrimination"]. A memory bank consists of the representations of all samples in the dataset. The dictionary for each mini-batch is randomly sampled from the memory bank with no back-propagation, so it can support a large dictionary size. However, the representation of a sample in the memory bank was updated when it was last seen, so the sampled keys are essentially about the encoders at multiple different steps all over the past epoch and thus are less consistent.

3.3. Pretext Task

Contrastive learning can drive a variety of pretext tasks. As the focus of this paper is not on designing a new pretext task, we use a simple one mainly following the *instance discrimination task* (Z. Wu, 2018, "Unsupervised feature learning via non-parametric instance discrimination").

We consider a query and a key as a positive pair if they originate from the same image, and otherwise as a negative sample pair. We take two random "views" of the same image under random data augmentation to form a positive pair. The queries and keys are respectively encoded by their encoders, f_q

and f_k

.

Algorithm 1 provides the pseudo-code of MoCo for this pretext task. For the current mini-batch, we encode the queries and their corresponding keys, which form the positive sample pairs. The negative samples are from the queue.

Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
   x_q = aug(x) # a randomly augmented version
   x_k = aug(x) # another randomly augmented version

   q = f_q.forward(x_q) # queries: NxC
   k = f_k.forward(x_k) # keys: NxC
   k = k.detach() # no gradient to keys

   # positive logits: Nx1
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

   # negative logits: NxK
   l_neg = mm(q.view(N,C), queue.view(C,K))

   # logits: Nx(1+K)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss, Eqn.(1)
   labels = zeros(N) # positives are the 0-th
   loss = CrossEntropyLoss(logits/t, labels)

   # SGD update: query network
   loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

   # update dictionary
   enqueue(queue, k) # enqueue the current minibatch
   dequeue(queue) # dequeue the earliest minibatch
```

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

Easy to understand

Technical details

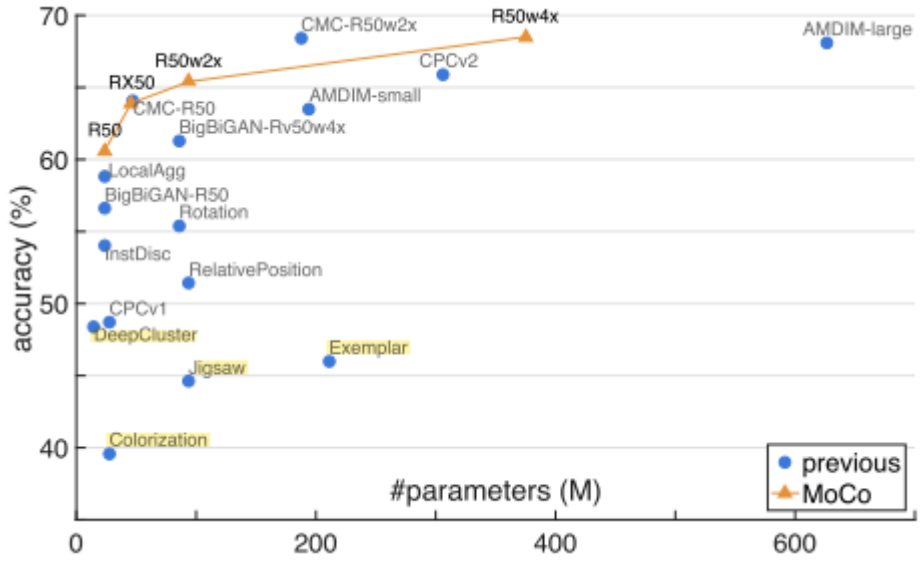
ResNet is used as the encoder (feature extractor) and its last layer is a global average pooling layer. After that, it's followed by a fixed-dimensional output (128-D). This output vector is normalized by its L2-norm.

BN deteriorates the representation learning

In experiments, we found that using BN prevents the model from learning good representations, as similarly reported in [O. Henaff, "Data-efficient image recognition with contrastive predictive coding"] where they avoided using BN. The model appears to "cheat" the pretext task and easily finds a low-loss solution. This is possibly because the intra-batch communication among samples (caused by BN) leaks information.

In this paper, they solved the problem by shuffling the sample order in the current mini-batch before distributing it among GPUs (and shuffle back after encoding).

4. Results



Comparison under the linear classification protocol on ImageNet.

T. Chen et al., 2020, "A simple framework for contrastive learning of visual representations" (SimCLR)

SimCLR: a simple framework for contrastive learning of visual representations. We simplify recently proposed contrastive self-supervised learning algorithms without requiring specialized architectures or a memory bank. we systematically study the major components of the SimCLR framework. We show that

1. Composition of data augmentations plays a critical role in defining effective predictive tasks that yield effective representations.
2. Using the *MLP projection head* substantially improves the quality of the learned representations.
3. Representation learning with contrastive cross-entropy loss (i.e., InfoNCE) benefits from L2-normalized embedding and an appropriately adjusted temperature parameter.
4. Contrastive learning benefits from larger batch sizes and more training steps compared to supervised learning. (Yet, you have to be aware that SimCLR requires a large batch size to obtain a large number of negative pairs since it takes the [end-to-end learning] without a memory bank. The large batch size is not really affordable for "ordinary" devices by memory-wise).

2. Method

2.1. The Contrastive Learning Framework

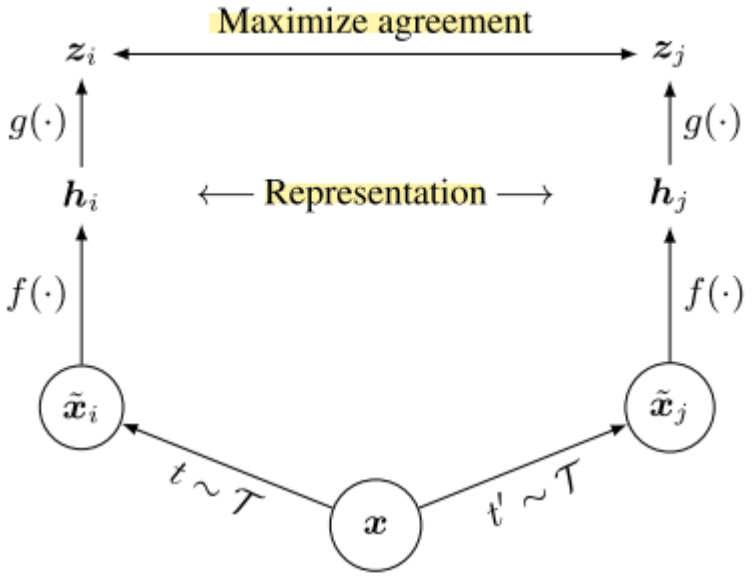


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

This SimCLR framework comprises the following major components:

1. We sequentially apply three simple augmentations: random cropping followed by resize back to the original size, random color distortions, and random Gaussian blur.
2. ResNet is adopted for the encoder f(.)

to obtain $h_i=f(\hat{x}_i)$. where $h_i \in R_d$ is the output after the average pooling layer.

two layers of MLP projection head g(.)

1. is used that maps a 2048-dimensional feature to a 128-dimensional latent space.
2. The same contrastive loss used in [Z. Wu, 2018, "Unsupervised feature learning via non-parametric instance discrimination"] and [A. Oord, 2018, "Representation learning with contrastive predictive coding"] is used.

We randomly sample a minibatch of N

examples and define the contrastive prediction task on pairs of augmented examples derived from the minibatch, resulting in 2N data points. We do not sample negative

examples explicitly. INstead, given a positive pair, we treat the other $2(N-1)$ augmented examples within a minibatch as negative examples. Let $\text{sim}(u,v)=u^Tv/||u||||v||$ denote the dot product between l_2 normalized u and v (i.e. cosine similarity). Then, hte loss function for a positiv pair of examples (i,j)

is defined as

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \qquad (1)$$

where $1_{[k\neq i]} \in \{0,1\}$

is an indicator function evaluating to 1 if $k\neq i$ and τ denotes a temperature parameter. The final loss is computed across all positive pairs, both (i,j) and (j,i)

, in a mini-batch.

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

Note that loss is symmetrical between two views.

2.2. Training with Large Batch Size

To keep it simple, we do not train the model with a memory bank. Instead, we use a large bath size. To support the large batch size, the authors used Cloud TPUs, using 32 to 128 cores depending on the batch size. SimCLR is not really efficient in terms of memory.

Global BN

In distributed training with data parallelism, the BN mean and variance are typically aggregated locally per device. However, the BN tends to deteriorate the quality of learned representations. We address this issue by aggregating BN mean and variance over all devices during the training. Other approaches include shuffling data examples across devices (MoCo), or replacing BN with layer norm (Henaff et al., 2019).

Default setting

We use learning rate warmup (linear warmup) for the first 10 epochs, and decay the learning rate with the cosine decay schedule without restarts.

Results

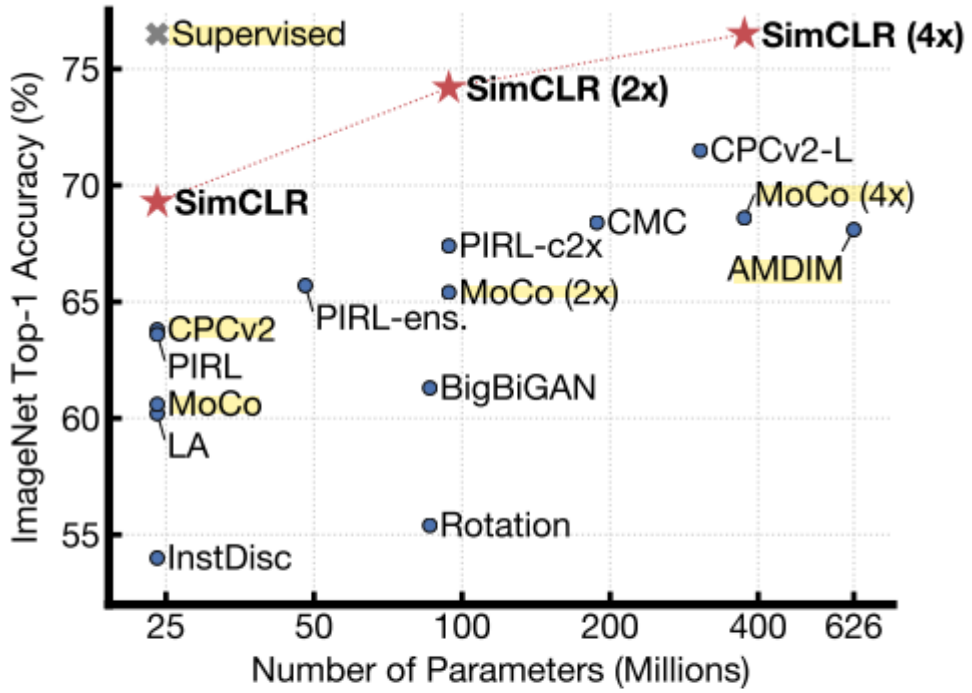


Figure 1. ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold.

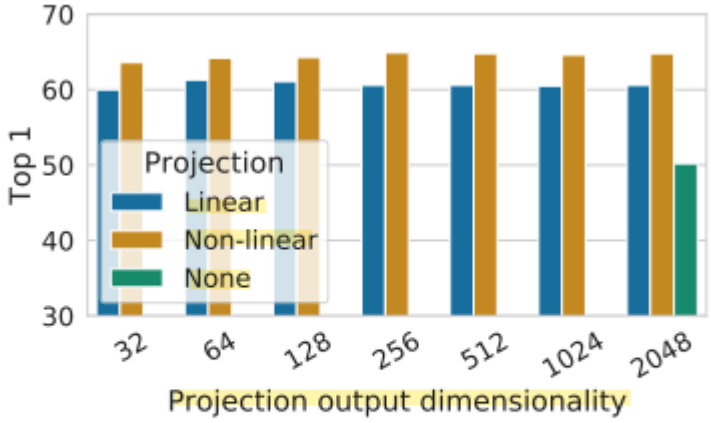


Figure 8. Linear evaluation of representations with different projection heads $g(\cdot)$ and various dimensions of $\boldsymbol{z} = g(\boldsymbol{h})$. The representation $\boldsymbol{h}$ (before projection) is 2048-dimensional here.

(Right) Use of the MLP projection head matters.

ℓ_2 norm?	τ	Entropy	Contrastive acc.	Top 1
Yes	0.05	1.0	90.5	59.7
	0.1	4.5	87.8	64.4
	0.5	8.2	68.2	60.7
	1	8.3	59.1	58.0
No	10	0.5	91.7	57.2
	100	0.5	92.1	57.0

Table 5. Linear evaluation for models trained with different choices of ℓ_2 norm and temperature τ for NT-Xent loss. The contrastive distribution is over 4096 examples.

τ seems to result in a good performance when it's 0.1 or 0.2 (considering the SimCLR or MoCo-v2 papers). Note that Entropy denotes data impurity.

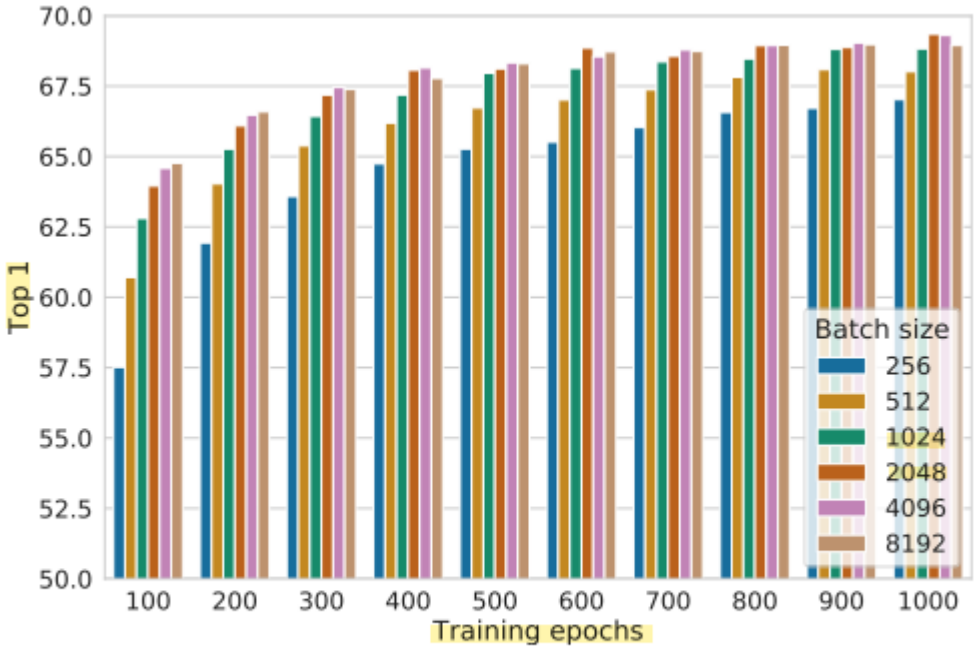


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.[10]

Large batch size results in the better performance (i.e., more negative pairs). Yet, it's not really affordable memory-wise.

X. Chen et al., 2020, "Improved baselines with momentum contrastive learning" (MoCo v2)

Introduction

The authors adopt two of SimCLR's design improvements to upgrade MoCo-v1 to MoCo-v2: (1) MLP projection head, (2) Stronger data augmentation.

The MoCo framework can process a large set of negative samples without requiring large training batches as shown in the figure below. In contrast to SimCLR's large 4k~8k batches which require TPU support, the "MoCo-v2" can run on a typical 8-GPU machine and achieve better results than SimCLR.

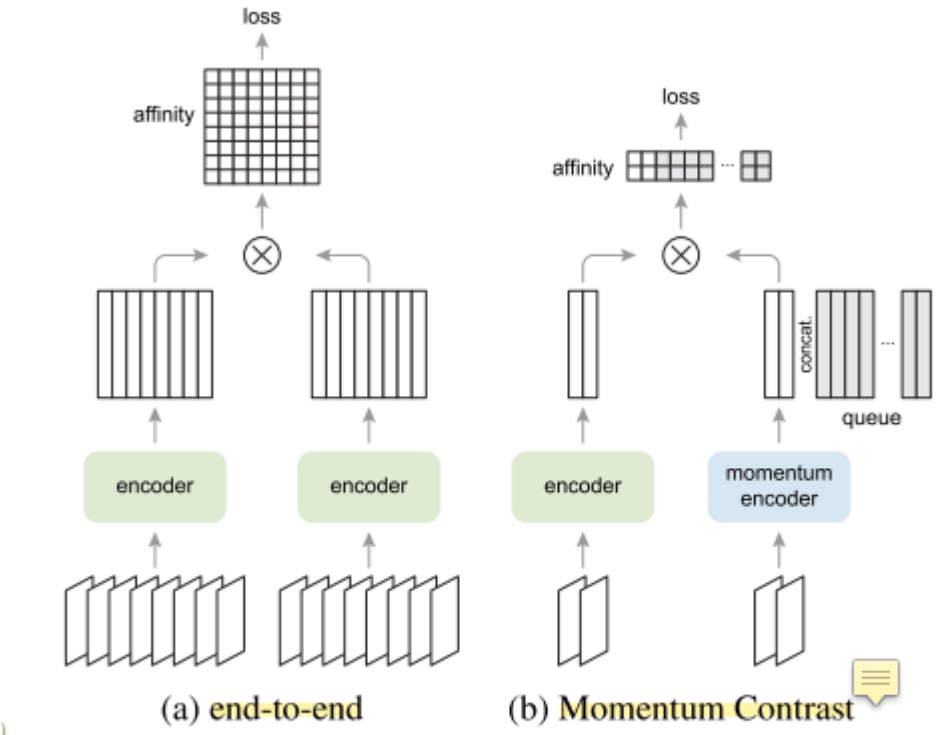


Figure 1. A **batching** perspective of two optimization mechanisms for contrastive learning. Images are encoded into a representation space, in which pairwise affinities are computed.

SimCLR follows Fig. 1(a); MoCo follows Fig.1(b).

Results

case	unsup. pre-train MLP	aug+	cos	epochs	ImageNet acc.	VOC detection AP_{50}	AP	AP_{75}
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	**82.5**	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	**800**	**71.1**	**82.5**	**57.4**	**64.0**

Table 1. **Ablation of MoCo baselines**, evaluated by ResNet-50 for (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). "**MLP**": with an MLP head; "**aug+**": with extra blur augmentation; "**cos**": cosine learning rate schedule.

case	unsup. pre-train MLP	aug+	cos	epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
*results of **longer** unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Table 2. **MoCo *vs.* SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. "aug+" in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

The MLP projection head improves the performance a lot. Longer epochs should be noted too. Yet, the cosine learning rate schedule is not much of a help.

mechanism	batch	memory / GPU	time / 200-ep.
MoCo	256	**5.0G**	**53 hrs**
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G†	n/a

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.

The 4k batch sie is intractable even in a high-end 8-GPU machine. Also, under the same batch ie of 256, the end-to-end variant is still more costly in memory and time, because it back-propagates to both q and k encoders, while MoCo back-propagates to the q encoder only.

J. Grill et al., 2020, "Bootstrap Your Own Latent A New Approach to Self-Bootstrap Your Own Latent A New Approach to Self-Supervised Learning" (BYOL)

BYOL relies on two neural networks, referred to as *online* and *target* networks. From an augmented view of an image, we train the online network to predict the target network representation of the same image under a different augmented view. At the same time, we update the target network with a slow-moving average of the online network. While SOTA methods rely on negative pairs, BYOL achieves a new SOTA *with them*.

1. Introduction

The contrastive learning-based approaches with negative pairs need careful treatment of negative pairs by either relying on large batch sizes, memory banks.

BYOL achieves higher performance than SOTA contrastive methods without using negative pairs. It iteratively bootstraps (the term bootstrap here is used in its idiomatic sense rather than the statistical sense.) the outputs of a network to serve as targets for an enhanced representation. Moreover, BYOL is more robust to the choice of image augmentations than contrastive methods. We suspect that no relying on negative pairs is one of the leading reasons for its improved robustness. Starting from an augmented view of an image, BYOL trains its online network to predict the target network's representation of another augmented view of the same image. While this objective admits collapsed solutions, e.g., outputting the same vector for all images, we empirically show that BOYL does not converge to such solutions.

We show that BOYL is more resilient to changes in the batch size and in the set of image augmentations compared to its contrastive counterparts.

2. Related work

The idea of a slow-moving average target network to produce stable targets for the online network was inspired by deep RL such as DQN.

Difference between SimCLR and BYOL w.r.t the slow-moving average network: In self-supervised learning, MoCo uses a slow-moving average network (momentum encoder)to maintain consistent representations of negative pairs drawn from a memory bank. INstead, BOYL uses a moving average network to produce prediction targets as a means of stabilizing the bootstrap step.

3. Method

Many successful self-supervised learning approaches build upon the cross-view prediction framework. Typically, these approaches learn representations by predicting different views (e.g. different random crops) of the same image from one another. Many such approaches cast the prediction problem directly in representation space. However, predicting directly in representation space can lead to collapsed representations. Contrastive methods circumvent

this problem by reformulating the prediction problem into one of discrimination: from the representation of an augmented view, they learn to discriminate between the representation of another augmented view of the same image, and the representations of augmented views of different images. Yet, this discriminative approach typically requires comparing each representation of an augmented view with many negative examples, to find ones sufficiently close to make the discrimination task challenging.

In case of BYOL: from a given representation, referred to as *target*, we can train a new, potentially enhanced representation, referred to as *online*, by predicting the target representation. From there, we can expect to build a sequence of representations of increasing quality by iterating this procedure, using subsequent online networks as new target networks for further training.

3.1. Description of BYOL

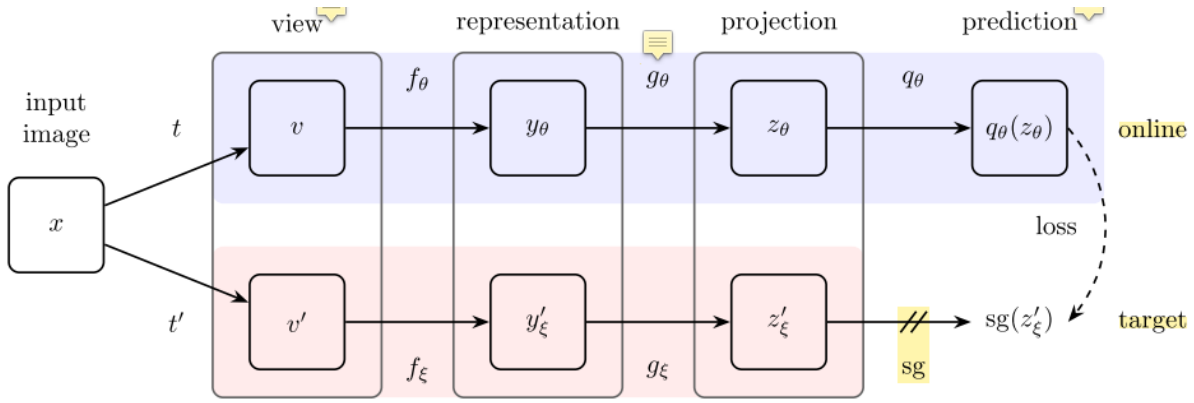


Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the image representation.

BYOL's goal is to learn a representation y_θ

which can then be used for downstream tasks. The online network is defined by a set of θ and is comprised of three stages: an encoder f_θ, a projector g_θ, and a predictor q_θ. The target network has the same architecture as the online network, but uses a different set of weights ξ. The target network provides the regression targets to train the online network, and its parameters ξ are an exponential moving average of the online parameters θ. More precisely, given a target decay rate $\tau \in [0,1]$

, after each training step, we perform the following update,

$$\xi \leftarrow \tau\xi + (1-\tau)\theta$$

We l_2

-normalize $q_\theta(z_\theta)$ and z'_ξ and yield $\bar{q}_\theta$ and $\bar{z}_{\xi'}$

. Note that this predictor is only applied to the online branch, making the architecture asymmetric between the online and target pipeline. Finally, we define the following mean squared error between the normalized predictions and target projections:

$$\mathcal{L}_{\theta,\xi} \triangleq \left\| \overline{q_\theta}(z_\theta) - \overline{z}'_\xi \right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\left\| q_\theta(z_\theta) \right\|_2 \cdot \left\| z'_\xi \right\|_2}. \tag{2}$$

We symmetrize the loss $L_{\theta,\xi}$

b separately feeding v′ to the online network and v to the target network to compute $\tilde{L}_{\theta,\xi}$. At each training step, we perform a stochastic optimization step to minimize $L^{BYOL}_{\theta,\xi} = L_{\theta,\xi} + \tilde{L}_{\theta,\xi}$ with respect to θ only, but not ξ

, as depicted by the stop-gradient (sg) in Figure 2. BYOL's dynamics are summarized as:

$$\theta \leftarrow \text{optimizer}\left(\theta, \nabla_\theta \mathcal{L}^{\text{BYOL}}_{\theta,\xi}, \eta\right), \tag{3}$$
$$\xi \leftarrow \tau\xi + (1-\tau)\theta, \tag{1}$$

where η

is a learning rate.

At the end of training, we only keep the encoder \fθ

as in the MoCo paper.

3.2 Intuitions on BYOL's behavior

As BYOL does not use an explicit erm to prevent collapse (such as negative pairs) while minimizing $L^{BYOL}_{\theta,\xi}$

with respect to θ, it may seem that BYOL should converge to a minimum of this loss with respect to (θ,ξ) (e.g., a collapsed constant representation). However, BOYL's target parameters ξ updates are **not** in the direction of $\nabla_\xi L^{BYOL}_{\theta,\xi}$. More generally, we hypothesize that there is no loss $L_{\theta,\xi}$ such that BYOL's dynamics is a gradient descent on L jointly over θ,ξ

. This way, the two representations are not forced to be the same (=> no worries for the collapsed constant representation).

When assuming BOYL's predictor q_θ

to be optimal i.e., $q_\theta = q_*$

with

$$q^\star \triangleq \arg\min_q \mathbb{E}\left[\left\| q(z_\theta) - z'_\xi \right\|_2^2\right], \quad \text{where} \quad q^\star(z_\theta) = \mathbb{E}\left[z'_\xi | z_\theta\right], \tag{4}$$

3.3 Implementation details

Architecture

The representation y

corresponds to the output of the final average pooling layer, which has a feature dimension of 2048. The representation y is projected to a smaller space by an MLP g_θ, and similarly for the target projection g_ξ

. This MLP consists of a linear layer with output size 4096 followed by BN and ReLU, and a final linear layer with output dimension 256.

4. Results

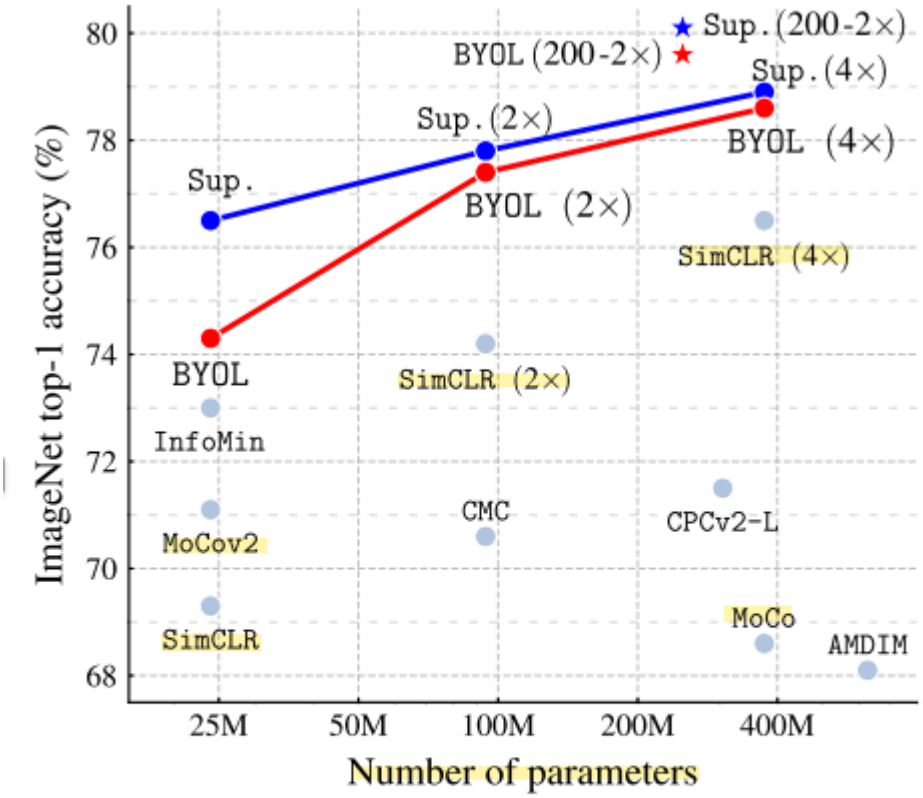
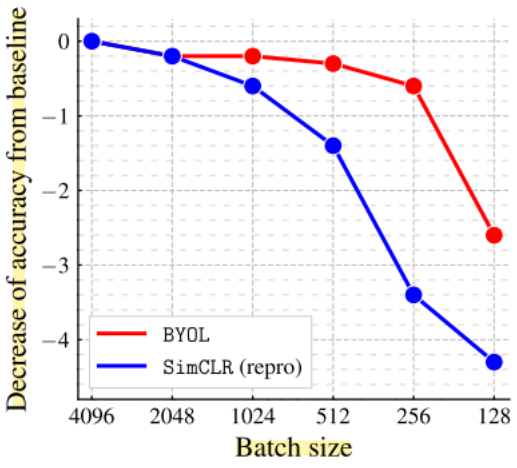
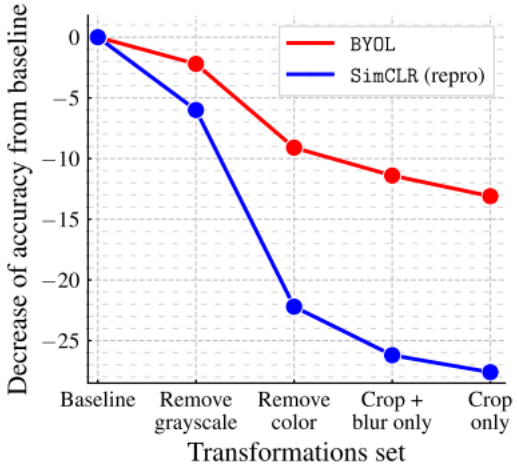


Figure 1: Performance of BYOL on ImageNet (linear evaluation) using ResNet-50 and our best architecture ResNet-200 (2×), compared to other unsupervised and supervised (Sup.) baselines [8].



(a) Impact of batch size



(b) Impact of progressively removing transformations

Figure 3: Decrease in top-1 accuracy (in % points) of BYOL and our own reproduction of SimCLR at 300 epochs, under linear evaluation on ImageNet.

(Left) It shows that BOYL is robust in terms of batch size, unlike SimCLR, (Right) BYOL is less affected by the image augmentations.

Target	τ_{base}	Top-1
Constant random network	1	$18.8_{\pm 0.7}$
Moving average of online	0.999	69.8
Moving average of online	0.99	**72.5**
Moving average of online	0.9	68.4
Stop gradient of online†	0	0.3

Ablation at 300 epochs under liner evaluation on ImagNet. The target decay rate should be appropriately set.

Weight decay: We note that removing the weight decay in either BYOL or SimCLR leads to network divergence, emphasizing the need for weight regularization in a self-supervised setting. In the SimCLR and BYOL frameworks, the weight decay of 10^{-6}

is used.

X Chen et al., 2020, "Exploring simple siamese representation learning" (Simple Siamese)

H. Liu et al., 2020, "FROST: Faster and more robust one-shot semi-supervised training"

Two previous SOTA methods that have been successful for one-shot semi-supervised learning

FixMatch and BOSS

FixMatch was the first paper to present results for one-shot semi-supervised learning, which was followed up by the BOSS method.

Other semi-supervised methods

Other semi-supervised methods such as the *π*

model, temporal ensembling, VAT, ICT, UDA, S4L

, Mix-Match, and Mean Teachers could not demonstrate results with less than 25 labeled examples per class.

Semi-supervised learning

Semi-supervised learning uses a small amount of labeled data to define the desired task and uses a large amount of unlabeled data to avoid overfitting the labeled data.

Consistency regularization (= consistency training) v.s Contrastive learning

It utilizes unlabeled data by relying on the assumption that the model should output these same final predictions when fed perturbed versions as on the original image. In contrastive representation learning, the assumption is that the model's internal representations should be similar when fed perturbed versions as on the original image and these representations should be less similar than the representation of other images.

BOSS - Class balancing ★

BOSS borrowed techniques from the literature on training with imbalanced data (i.e. some classes having many more training samples than other classes). An important difference of the semi-supervised learning from the data imbalanced domains is the lack of ground truth as to what are the majority and minority classes when applying these techniques to unlabeled data. The authors of BOSS proposed using the model's pseudo-labels as a means to count each class and implicitly assumed that the unlabeled dataset is class balanced.

Class balancing methods are typically one of two types: 1) data-level, 2) algorithm-level. The *data-level methods* oversample the minority classes and undersample the majority classes. The *algorithm-level methods* improve the imbalance by using larger weights in the loss function for training samples that belong to the minority classes, as well as smaller weights for the samples from the majority classes.

Then, the count of the set of pseudo-labels was used to conduct the class balancing. The BOSS includes the data-level, algorithm-level, and hybrid methods.

Pseudo-labeling

It defines a base model that is trained on labeled data and the model is used to predict labels for unlabeled data.

Self-training

It is a basic approach of pseudo-labeling where a classifier is trained with an initially small number of labeled examples, aiming to classify unlabeled points, and then it is retrained with its own most confident predictions, thus enlarging its labeled training set.

How is FROST different from the previous semi-supervised learning methods?

FROST differs from the previous methods by utilizing a semi-supervised base learner in place of the supervised base learning.

Loss function in FROST

A weighted sum of a supervised loss and an unsupervised loss is used, in which the unsupervised loss term is for consistency regularization.

$J=\lambda_s J_s+\lambda_u J_u$

where the subscripts s

and u denote supervised and unsupervised, respectively. λ and J

denote a scalar hyperparameter and loss, respectively.

Xinlei Chen and Kaiming He, 2020, Exploring Simple Siamese Representation Learning (SimSiam)

Collapsed representation

It refers to the phenomenon that all representations collapse into a constant. To solve this problem, contrastive learning was initially developed.

Application of Siamese networks

The applications include signature and face verification, tracking, one-shot learning, and others.

Negative samples in contrastive learning

In practice, contrastive learning methods benefit from a large number of negative samples. These samples can be maintained in a memory bank. In a Siamese network, *MoCo* [Kaiming He et al., 2019, Momentum Contrast for Unsupervised Visual Representation Learning] maintains a queue of negative samples and turns one branch into a momentum encoder to improve consistency of the queue. *SimCLR* directly uses negative samples coexisting in the current batch, and it requires a large batch size to work well.

Common experimental setup protocol for unsupervised learning

We do unsupervised pre-training on the (1000-class ImageNet) training set without using labels. The quality of the pre-trained representations is evaluated by training a supervised linear classifier on frozen representations in the training set, and then testing it in the validation set which is a common protocol.

kNN classifier serving as a monitor of the progress

Originally, this idea is from the memory bank paper:

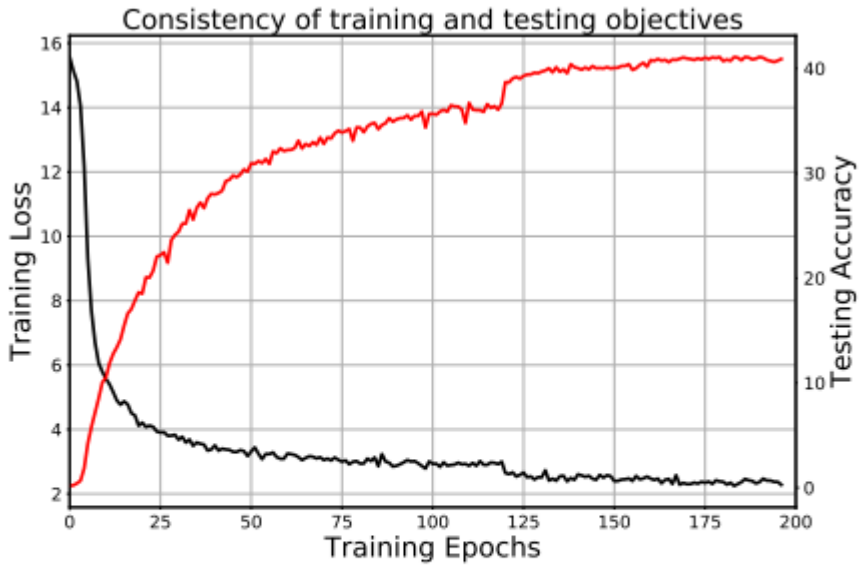


Figure 4: Our kNN testing accuracy on ImageNet continues to improve as the training loss decreases, demonstrating that our unsupervised learning objective captures apparent similarity which aligns well with the semantic annotation of the data.

In the ImageNet dataset, all the images have labels. We just train a model without labels for unsupervised learning. The above figure can be created by finding neighboring representations nearby a test image's representation and compute:

#matchedlabelsk/k

where k

is a number of neighbors considered.

Categories of unsupervised learning for the representation learning

Unsupervised learning for representation learning is categorized into 1) Generative, 2) Discriminative.

Generative approaches to representation learning build a distribution over data and latent embedding and use the learned embeddings as image representations. Many of these approaches rely either on auto-encoding of images or on adversarial learning. The generative methods typically operate directly in pixel space. This, however, is computationally expensive, and the high level of detail required for image generation may not be necessary for representation learning.

Among the *discriminative methods*, contrastive methods currently achieve SOTA performance in self-supervised learning. The contrastive approaches avoid a costly generation step in pixel space by bringing representation of different views of the same image closer ('positive pairs'), and spreading representations of views from different images

('negative pairs') apart. The contrastive methods often require comparing each example with many other examples to work well.

Some self-supervised methods are not contrastive but rely on using auxiliary handcrafted prediction tasks to learn their representation (i.e. *pretext tasks*). In particular, relative path prediction, colorizing gray-scale images, image inpainting, image jigsaw puzzle, image super-resolution, and geometric transformations have been shown to be useful.

Transfer to other classification tasks

In the evaluation step, the authors evaluated the representation on other classification datasets to assess whether the features learned on ImageNet are generic and thus useful across image domains. They performed the linear evaluation and fine-tuning on the same set of classification tasks. The resulting table from the paper is shown in the following figure:

Method	Food101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
BYOL (ours)	**75.3**	91.3	**78.4**	**57.2**	**62.2**	**67.8**	60.6	82.5	75.5	90.4	94.2	**96.1**
SimCLR (repro)	72.8	90.5	74.4	42.4	60.6	49.3	49.8	81.4	**75.7**	84.6	89.3	92.6
SimCLR [8]	68.4	90.6	71.6	37.4	58.8	50.3	50.3	80.5	74.5	83.6	90.3	91.2
Supervised-IN [8]	72.3	**93.6**	78.3	53.7	61.9	66.7	**61.0**	**82.8**	74.9	**91.5**	**94.5**	94.7
Fine-tuned:												
BYOL (ours)	**88.5**	**97.8**	86.1	**76.3**	63.7	91.6	**88.1**	**85.4**	**76.2**	91.7	**93.8**	97.0
SimCLR (repro)	87.5	97.4	85.3	75.0	63.9	91.4	87.6	84.5	75.4	89.4	91.7	96.6
SimCLR [8]	88.2	97.7	85.9	75.9	63.5	91.3	88.1	84.1	73.2	89.2	92.1	97.0
Supervised-IN [8]	88.3	97.5	**86.4**	75.8	**64.3**	**92.1**	86.0	85.0	74.6	**92.1**	93.3	**97.6**
Random init [8]	86.9	95.9	80.2	76.1	53.6	91.4	85.9	67.3	64.8	81.5	72.6	92.0

Table 3: Transfer learning results from ImageNet (IN) with the standard ResNet-50 architecture.