Self-Supervised Learning

1. Contrastive Learning

Discriminative approaches learn representation by using objective functions that train networks on pretext tasks that have labels derived from an unlabeled dataset.

1.1. SimCLR

No need for specialized architectures or memory banks.

- Composition of multiple data augmentations is crucial for defining the contrastive prediction tasks.
- A learnable non-linear transform between the representation and contrastive loss improves the quality of representations learned.
- If contrastive cross-entropy is used, normalized embeddings and tuned tempreture is desirable.
- Larger batch-size with longer training.

1.1.1. Method

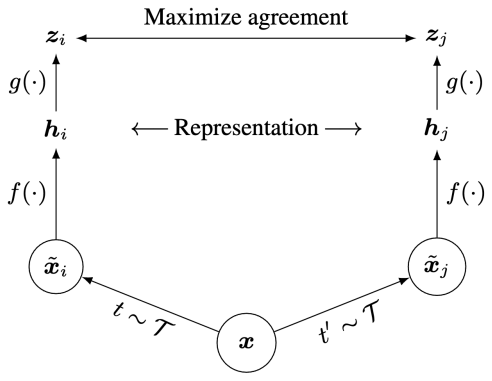
Create two correlated views from the same example by using data augmentations. These views form a positive pair $\tilde{x}_i, \tilde{x}_j$.

The base encoder $f(\cdot)$ extracts representations from examples to get $\boldsymbol{h}_i = f(\tilde{x}_i)$ where $\boldsymbol{h}_i \in \mathbb{R}^d$ is the output after the average pooling layer.

The projection head $g(\cdot)$ maps representations to the contrastive-loss space. $\boldsymbol{z_i} = h(\boldsymbol{h}_i)$

The contrastive loss function is defined so that given a set $\{\tilde{x}_k\}$ including a positive pair, the task aims to identify $\tilde{x}_j \in \{\tilde{x}_k\}_{k \neq i}$ for a given $\tilde{x}_i$.

$$l_{i,j} = -\log \frac{\exp\big(\text{sim}\big(\boldsymbol{z}_i, \boldsymbol{z}_j\big)/\tau\big)}{\sum_{k=1,k\neq i}^{2N} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$



1.2. MoCo

Methods using the contrastive loss can be thought of as building dynamic dictionaries. The keys in the dictionary are sampled from data and are represented by an encoder network. Unsupervised learning trains encoders to perform dicitonary look-up: an encoded query should be similar to its matching key and dissimilar to others.

Thus build a large and consistent dictionary through the training. Keys should be represented by a similar encoder so that thie comparisons to the query are consistent.

A queue of data samples is maintained, new encodings are queued and old ones are dequeued. As a results dictionary size is independent of the batch size. A slowly moving key-encoder is implemeted as a momentum-based moving average of the query-encoder.

1.2.1. Method

Consider an encoded query q and a set of encoded samples $\{k_0, k_1, k_2, ...\}$ that are keys of a dictionary. Assume one of the keys, k_+ matches q. Loss is low when q is similar to k_+ and dissimilar to all other keys. This loss is the log loss of a $(K+1)$-way softmax-based classifier that tries to classify q as k_+.

Using a queue can make the dictionary large but it also makes it intractable to update the key encoder by back-propagation. As a result key encoder's weights are updated by:

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

Here, $m \in [0, 1)$ is a momentum coefficient. Only the parameters of the query-encoder is updated by back-propagation. As a result, though the keys in the queue are encoded by different encoders (in different batches), the differnence among them can be made small.

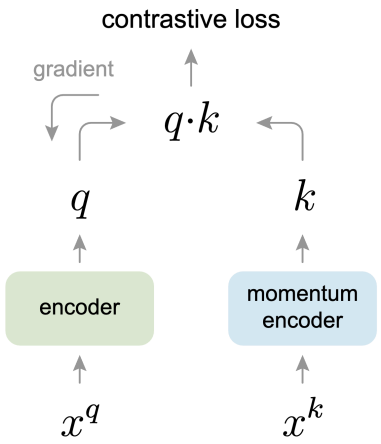


Figure 1: Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys {k0, k1, k2, ...} are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current 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."

2. Distillation

2.1. BYOL

State-of-the-art contrastive methods are trained by reducing the distance between representations of different augmented views of the same image ("positive pairs"), and increasing the distance between representations of augmented views from different images ("negative pairs").

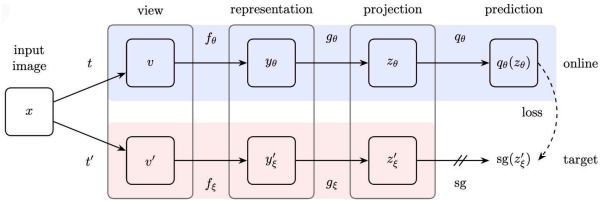
While previous methods based on bootstrapping have used pseudo-labels, cluster indices or a handful of labels, we propose to directly bootstrap the representations. In particular, BYOL uses two neural networks, referred to as online and target networks, that interact and learn from each other. Starting 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 BYOL does not converge to such solutions.

DeepCluster partially answers the question of wheter or not negative pairs are necessary. It uses bootstrapping on previous versions of its representation to produce targets for the next representation; it clusters data points using the prior representation, and uses the cluster index of each sample as a classification target for the new representation. While avoiding the use of negative pairs, this requires a costly clustering phase and specific precautions to avoid collapsing to trivial solutions.

BYOL introduces an additional predictor on top of the online network, which prevents collapse. BYOL uses a moving average network to produce prediction targets as a means of stabilizing the bootstrap step.

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

The online network is defined by a set of weights 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.



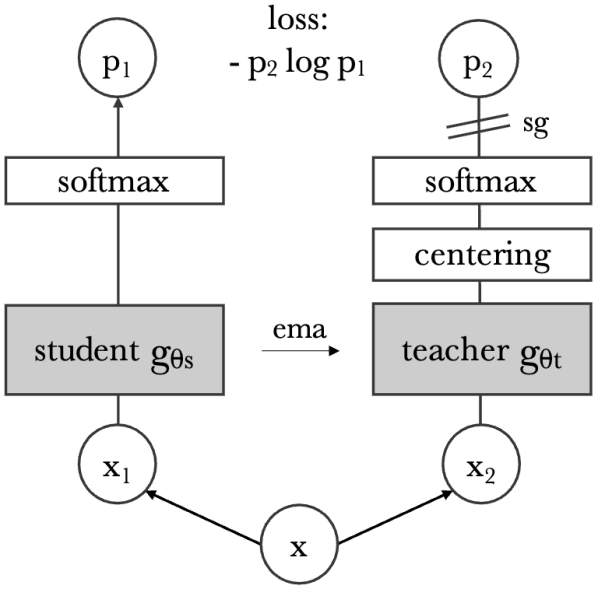
As BYOL does not use an explicit term to prevent collapse (such as negative examples) while minimizing $\mathcal{L}^{\mathrm{BYOL}}_{\theta,\xi}$ with respect to θ, it may seem that BYOL should converge to a minimum of this loss with respect to (θ, ξ) (e.g., a collapsed constant representation). However BYOL's target parameters ξ updates are not in the direction of $\nabla_\xi \mathcal{L}^{\mathrm{BYOL}}_{\theta,\xi}$. More generally, we hypothesize that there is no loss $\mathcal{L}_{(\theta,\xi}$ such that BYOL's dynamics is a gradient descent on $\mathcal{L}$ jointly over θ, ξ. This is similar to GANs,

where there is no loss that is jointly minimized w.r.t. both the discriminator and generator parameters. There is therefore no a priori reason why BYOL's parameters would converge to a minimum of $\mathcal{L}^{\text{BYOL}}_{\theta,\xi}$.

2.2. DINO

- Self-supervised ViT features explicitly contain the scene layout and object boundaries. This information is directly accesible in the self-attention modules of the last block.
- Self-supervised ViT features perform particularly well with a basic nearest neighbors classifier without any finetuning, linear classifier nor data augmentation.
- Smaller patches need to be used in order to improve the qualty of features.

DINO, directly predicts the output of a teacher network, built with a momentum encoder, by using a standart cross-entropy loss. Method can work only with centering and sharpening of the teacher output to prevent collapse.



Knowledge distillation is a learning paradigm where a student network g_{θ_s} is trained to match the output of a given teacher network g_{θ_t}, parameterized by θ_s and θ_t respectively. Given an input image x, both networks output probability distributions over K dimensions denoted by P_s and P_t. the probability P is obtained by normalizing the output of the network g with a softmax function.

$$P_s(x)^{(i)} = \frac{\exp\left(g_{\theta_s}(x)^{(i)}/\tau_s\right)}{\sum_{k=1}^{K}\exp\left(g_{\theta_s}(x)^{(k)}/\tau_s\right)}$$

τ_s is a tempreture parameter that controls the sharpness of the output distribution, and a similar formula holds for P_t with tempreture τ_t.

$$\min_{\theta_s} H(P_t(x), P_s(x)), H(a,b) = a - \log b$$

Is the cross-entropy loss being minimized with respect to θ_s in order to match the distributions.

Different distorted views or crops of an image is generated. A total V different views including two global views x_1^g, x_2^g and several local views of smaller resolution. All crops are passed through the student while only global views are passed through the teacher.

$$\min_{\theta_s} \sum_{x\in\{x_1^g, x_2^g\}} \sum_{\substack{x'\in V \\ x'\notin x}} H(P_t(x), P_s(x'))$$

Here local views in V are of size $(96, 96)$ and global views in V are of size $(224, 224)$.

Student and teacher are the same architecture g with a different set of parameters θ_s and θ_t. θ_s is learned by minimizing the above equation. Since there is no a priori teacher g_{θ_t} it needs to built from the past iterations of the student network. Teacher network is updated with the rule $\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$, with lambda following a cosine scheduling regime from 0.996 to 1. "We observe that this teacher has better performance than the student throughout the training, and hence, guides the training of the student by providing target features of higher quality. This dynamic was not observed in previous works"

ViT does not uae batch norms, thus DINO also doesn't use them.

To preevent collapse either contrastive loss, clustering constraints, predictor or batch normalizations are used. Even though it is possible to stabilize the framework with norms, centering and sharpening of the momentum teacher outputs work as well. Centering prevents one dimension to dominate but encourages collapse to the uniform distribution, while the sharpening has the opposite effect.

$$g_t(x) \leftarrow g_t(x) + c \text{ where,}$$

$$c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i)$$

Above equation describes the centering equation, sharpening is obtained by using a low value for the temperature τ_t in the teacher softmax normalization.

3. Clustering-based Methods

3.1. DeepCluster

The method consists of alternating between clustering of image descriptors and updating the weights of the network by predicting the cluster assignments. Requires little domain knowledge, no specific signal from the inputs.

A multilayer perpectron classifier on top of the last convolutional layer of a random AlexNet achieves 12% in accuracy on ImageNet, while the chance is at 0.1%. Seeing this one can use this innate property of convnets to get a rough representation and generate pseudo-labels by clustering the features.

3.1.1. How to avoid trivial solutions?

An optimal decision boundary is to assign all inputs to a single cluster. To solve "Feature quantization" can be used which automatically reassigns empty clusters by setting the empty centroid to a random non empty centroid plus a perturbation. The features belonging to the non empty centroid selected are split between the two.

If most features are assigned to a few clusters, with many singleton clusters; convnet predicts the same outputs regardless of input, trivial parameterization. To solve images are sampled based on a uniform distribution over classes, or pseudo-labels. Meaninig the loss contribution of an input is inversely proportional to its assigned classes' size.

3.2. DeeperCluster

Since clustering methods build supervision from inter-image similarities, the task at hand becomes inherently more complex when the number of images increases. In addition, DeepCluster captures finer relations between images when the number of clusters scales with the dataset size. Clustering approaches infer target labels at the same time as features are learned. Thus, target labels evolve during training, making clustering-based approaches unstable. Furthermore, these methods are sensitive to data distribution as they rely directly on cluster structure in the underlying data. Explicitly dealing with unbalanced category distribution might be a solution but it assumes that we know the distribution of the latent classes.

Self-supervised learning methods extract pseudo-labels from the input signal, meaning they levarage intra-image statistics which are often independent from the data distribution.

To combine they assume that the inputs $x_1, ..., x_N$ are rotated images each assocaited with a target-label y_n encoding its rotation angle and a cluster assignment z_n. The cluster assignment changes during training along with the visual representations. Say $\mathcal{Y}$ is the set of possible rotation angles and $\mathcal{Z}$ is the posible cluster assignments.

$$\min_{\theta, W} \sum_{n=1}^{N} \ell(y_n \times z_n, W f_\theta(x_n))$$

Since the cartesian product space can get very large with the number of clusters ($O(|\mathcal{Y}|\,|\mathcal{Z}|)$), a hierarchial loss is used that approximates the given loss is used. First we partition the target labels into S super-classes and we denote by y_n the super-class assignment vector in $\{0.1\}^S$ of the image n and by y_{ns} the sth entry of y_n. This assignemnt is done via a linear classifier V on top of features. Each super-class is further partitioned by z_n^s the vector in $\{0,1\}^{k_s}$ of the assignments into k_s sub-classes for an image n belonging to super-class s.

Every T epochs k-means is ran to get m centroids. Cartesian product is used to create the $4m$ super-classes, with 4 rotations. Then these clusters are further split into k sub-classes with k-means.

3.3. Self-Label

Learning a DNN together while discovering the data labels can be viewed as simultaneous clustering and representation learning. DeepCluster cannot be described as the optimization of an overall learning objective, instead there exist degenerate solutions that the algorithm avoids by particular implementation choices.

To stop mode collapse, a constraint is added that the labels must induce an equipartition of the data. This maximizes the information between data indices and labels. The resulting label assignment problem turns out to be optimal transport which can be solved in poly time. To save time Sinkhorn-Knopp is used to get an approximate solution.

Consider a DNN $\boldsymbol{x} = \boldsymbol{\Phi}(I)$ mapping data I to feature vectors $\boldsymbol{x} \in \mathbb{R}^D$. The model is trained on N data points $I_1, ..., I_N$ with labels $y_1, ..., y_N \in \{1, ...K\}$ drawn from a space of K possible labels. The representation is followed by a classification head $h : \mathbb{R}^D \to \mathbb{R}^K$, whose output is mapped to probbilities via softmax

$$p(y = \cdot\,|\boldsymbol{x}_i) = \text{softmax}(h(\boldsymbol{\Phi}(\boldsymbol{x}_i)))$$

Model and head parameters are learned by minimizing the average cross-entropy loss

$$E(p|y_1, ..., y_N) = \frac{1}{N}\sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i)$$

Training with this objective requires a labelled dataset. In fully unsupervised case, all points can be assigned to an arbitrary class to solve the problem, to adress this issue:

$$E(p, q) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{K} q(y|\boldsymbol{x}_i) \log p(y|\boldsymbol{x}_i)$$

where $q(y|\boldsymbol{x}_i) = \delta(y - y_i)$ we can see that this second equation is equivalent to the first. Optimazing q corresponds to reassigning the labels which leads to degeneracy. To avoid this, constraint that the label assignments must partition the data in equaly-sized subsets is introduced.

$$\min_{p,q} \text{subject to } \forall y : q(y|\boldsymbol{x}_i) \in \{0, 1\} \text{ and } \sum_{i=0}^{N} q(y|\boldsymbol{x}_i) = \frac{N}{K}$$

Meannig each data point $\boldsymbol{x}_i$ is assigned to exactly one label and that overall all data points are split uniformly among K classes.