Report 2: Self-Supervised Learning
A theoretical discussion of SSL techniques

Niccolò Arati
niccolo.arati@edu.unifi.it

Abstract

This report is a review and comparison of some of the most used self-supervised learning methods, both for representation learning and evaluation (in a classification task). After a brief introduction of SSL, three main methods are introduced with a theoretical overview: SimCLR, BYOL and Barlow Twins. Following this, some methods of SSL evaluation are presented and compared with each other and with supervised models performances, showing that Linear Probe is the most consistent out of the analyzed methods. Lastly, performances from some pretrained (with SSL techniques) common model architectures are compared, focusing on Visual Transformers.

1. Introduction

Supervised Learning relies on labeled datasets to train a model. Acquiring such datasets could be expensive and labor intensive, since it requires competent people with domain knowledge, limiting the paradigm scalability in many real-world scenarios where annotations are either difficult or impractical to obtain.
In contrast, **Self-Supervised methods** leverage the abundance of readily available unlabeled data. Their goal is to develop a backbone network capable of independently identify meaningful features or patterns within the raw data and learning a representation that can be applied across a variety of tasks.
This report is divided into 3 main parts:

- A first part (Sec. 2) with an introduction of standard SSL training and some indications regarding the management of an unlabeled dataset.
- A second part (Sec. 3) where three different self-supervised methods associated with three categories of SSL will be presented: SimCLR (contrastive learning), BYOL (distillation learning) and Barlow Twins (redundancy reduction learning).
- Lastly, a third final part (Sec. 4) with an evaluation of self-supervised models, made of three main techniques which will be compared to some supervised models, and a brief analysis on some of the SSL pretrained models from timm package.

2. Dataset setup and SiameseNet

According to the SSL paradigm, an effective feature representation of an image should capture its semantic information and remain invariant to augmentations and nuisance factors. Two different views of the same subject should yield similar, if not identical, representations. In Fig. 1 there is an example on what the outcome could be.
In order to obtain this, a custom dataset is required. Given an image and some specified data augmentations, it returns two different views of the same original image by applying the augmentations.

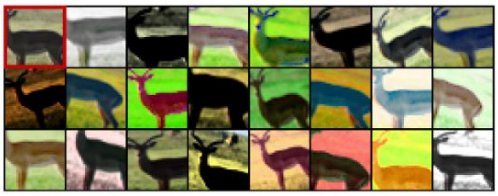

Figure 1. Example of images that should all be associated with a similar representation, because all of them contains comparable semantic information. All the possible couple of images could represent two different views of an original image in the custom dataset (some of the couples are more obvious to the human eye, while some other are not).

During the training process, a dataloader handles the custom dataset, and a single item of a batch is composed of the two augmented images that are given in input to a Siamese Network.
A **Siamese Network** is an architecture designed to learn and compare the similarity between two inputs (the two different views). It is composed of two neural networks (backbone) that works on different inputs, based on these two backbones there are two versions of a Siamese Network:

- **Symmetric**: both subnetworks are identical in architecture and share the same weights.

- **Asymmetric**: the subnetworks can have different architectures or weight configurations. They process inputs differently to account for differences in their structure or domain.

The extracted features should be as similar as possible if the two views belong to the same original image, otherwise the learned representations should be as different as possible. In Sec. 3, three different losses and/or architecture variations are presented in order to obtain this goal.
This report could be read in parallel with the code contained in this Gihtub repository: https://github.com/Sossio699/ComputerVision (in some of the following sections, some of the code will be exploited, but in general what is presented as a theoretical description in this report has a code counterpart).

3. Self-Supervised Methods

In this section, three self-supervised methods will be presented. All of them are based on the Siamese Network idea, but they employ different types of loss functions and slightly variations in the model architecture.
The goal is always the same: to learn a representation that is transformation-invariant and not costant for every single image. The second requirement is mandatory to avoid **representation collapse**, a common issue that occurs when the model maps all input data to the same or nearly identical representations, which then become uninformative.

3.1. SimCLR

SimCLR (Simple framework for Contrastive Learning of visual Representations [1]) is a SSL framework that uses **contrastive learning** to learn representations by maximizing similarity between augmented versions of the same image (positive pairs) and minimizing similarity with other images (negative pairs). This framework attempts to accomplish what was described above by adding a projector (usually a Multi-Layer Perceptron) on top of the Siamese Network backbone to map features to a lower dimensional space (latent space), where a contrastive loss will be applied.
This is the **contrastive loss** (InfoNCE) formula:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z_i}, \mathbf{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z_i}, \mathbf{z_k})/\tau)} \quad (1)$$

- $\mathbf{z_i}$ and $\mathbf{z_j}$ are the projected feature embeddings of a positive pair.
- $\text{sim}(\cdot)$ is the similarity function. If the features are normalized, this function is the cosine similarity: $\text{sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$
- τ is a temperature parameter controlling the sharpness of the similarity distribution.
- 2N is the total number of augmented views in the batch, since every image has two views.

As we can see in Fig. 2, a generic training step for a positive pair could be described with the original image $\mathbf{x}$ augmented in two different views, $\tilde{\mathbf{x}}_\mathbf{i}$ and $\tilde{\mathbf{x}}_\mathbf{j}$, and a forward pass through the Siamese Network encoders, which produces two representations $\mathbf{h_i}$ and $\mathbf{h_j}$. These learned representations are then fed to the projector, obtaining the latent representations $\mathbf{z_i}$ and $\mathbf{z_j}$. This process is applied for every image in the batch, and lastly the contrastive loss is calculated for each positive pair, with respect to the other images views that are considered as negative pairs ($\mathbf{z_i}$ and $\mathbf{z_k}$ in Eq. (1)).

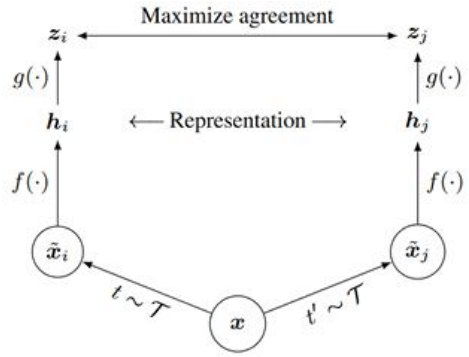

Figure 2. SimCLR architecture

After training, the projector will be discarded before fine-tuning on downstream tasks.
SimCLR usually requires strong data augmentation techniques to generate diverse views of the same image, while in supervised learning this could lead to a performance loss.
The importance of the learned representation prior to the nonlinear projection lies in the fact that the contrastive loss can lead to a loss of information. Specifically, $z = g(h)$ is optimized to ensure invariance to data transformations. Consequently, g may discard details that could be valuable for downstream tasks, such as object color or orientation. By applying the nonlinear transformation $g(\cdot)$, it becomes possible to preserve and structure more information in h.

3.2. BYOL

BYOL (Bootstrap Your Own Latent [2]) is a SSL algorithm that operates without using negative pairs. It uses two

networks, a target network and an online network, to bootstrap representations through distillation learning, relying on positive pairs created via data augmentations. The primary goal of BYOL is to learn a representation y_θ that can be used for downstream tasks.
As we can see from Fig. 3, the BYOL architecture is composed of:

- **Online Network**: defined by a set of weights θ, its architecture consists of an encoder f_θ, a projection head g_θ and a prediction head q_θ.
- **Target Network**: defined by a set of weights ξ, it shares the same architecture of the online network, except for the prediction head (the architecture is asymmetric).

The main passages into a training step are:

- Data Augmentation module: generates the positive pair v and v'.
- Encoder f: extracts features $y = f_\theta(v)$ and $y' = f_\xi(v')$ from input data.
- Projector g: maps the features into a latent representation space ($z = g_\theta(y)$ and $z' = g_\xi(y')$.
- Predictor q: predicts the target network's latent representations $p = q_\theta(z)$.

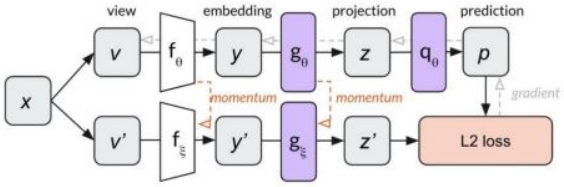

Figure 3. BYOL architecture

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.
The exponential moving average update is performed as:

$$\xi \leftarrow \tau\xi + (1-\tau)\theta \tag{2}$$

where $\tau \in [0, 1]$ is the target decay rate.
At the end of a training step, the L2 loss (mean squared error) between the normalized predictions and the target projections is computed as follows:

$$\mathcal{L}_{\theta,\xi}(x_1, x_2) = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta^{(1)}), z_\xi^{(2)} \rangle}{\|q_\theta^{(1)}\| \cdot \|z_\xi^{(2)}\|} \tag{3}$$

The loss is computed also for the couple (x_2, x_1), $\tilde{\mathcal{L}}_{\theta,\xi}$, and then a stochastic gradient descent step is performed in order to minimize $\mathcal{L}_{\theta,\xi}^{\text{BYOL}} = \mathcal{L}_{\theta,\xi} + \tilde{\mathcal{L}}_{\theta,\xi}$
$\mathcal{L}_{\theta,\xi}(x_1, x_2)$ and $\mathcal{L}_{\theta,\xi}(x_2, x_1)$ are both computed mainly for computational efficiency, but also because this enforces symmetry in learning, promotes representation invariance across views and helps stabilize training and reduce representaiton collapse. After the training, only the encoder f_θ will be used to accomplish the downstream task (by finetuning).
By using the projected representation of the target network as target for BYOL predictions, the weights of the target network represent a delayed and more stable version of the weights of the online network.
While contrastive learning methods benefit more from color augmentation, BYOL is in general more robust to the choice of data augmentation techniques.

3.3. Barlow Twins

Barlow Twins [3] is a SSL algorithm, its main innovation is achieved by promoting redundancy reduction between representations of different data augmentations. Barlow Twins encourages the embeddings of two augmented views of the same input to be similar while also being decorrelated across dimensions.
The goal of this technique is to learn representations that are both invariant to augmentations (similar representations for different augmentations of the same sample) and non-redundant (each feature in the representation space captures unique information).
As illustrated in Fig. 4, the architecture consists on a data augmentation module and a deep neural network with trainable parameters θ, generally consisting of an encoder and a projector.

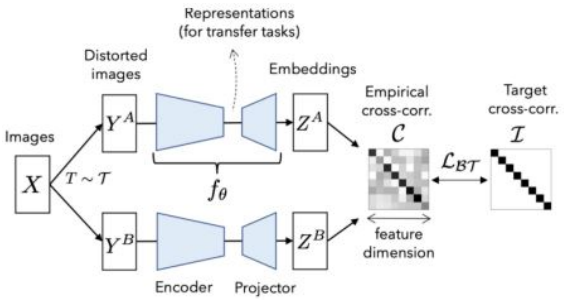

Figure 4. Barlow Twins architecture

A generic train step could be summed up as:

- From a batch X, obtain two views of each image, Y^A and Y^B, through data augmentation techniques.
- These two batch of views are fed to a function f_θ (shared deep neural network), producing the embeddings Z^A and Z^B.

- The **cross-correlation matrix** between the outputs of the two identical networks along the batch dimension is computed as:

$$C_{i,j} = \frac{\sum_b z^A_{b,i}, z^B_{b,j}}{\sqrt{\sum_b (z^A_{b,i})^2}\sqrt{\sum_b (z^B_{b,j})^2}} \quad (4)$$

where b indicates the batch samples and $z_{i,j}$ are elements of the networks' outputs.

- Compute the **Barlow Twins Loss**, defined as:

$$\mathcal{L}_{BT} = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2 \quad (5)$$

with λ positive constant.

- Backpropagation to update the weights θ with gradient descent to minimize the loss.

The goal is to make C as close as possible to the identity matrix. This ensures that the diagonal entries (C_{ii}) are close to 1, meaning each dimension in z is strongly correlated, and the off-diagonal entries (C_{ij}) are close to 0, reducing redundancy between different dimensions.
The first element of the Barlow Twins Loss (Eq. (5)) is the invariance term, which makes the embedding z invariant to the distortions applied (so there is a strong correlation between each of his dimensions). The second element of the loss is the redundancy reduction term, which decorrelates the different vector components of the embedding (by penalyzing the off-diagonal correlations).
Barlow Twins and SimCLR share the same network architecture, but they employ different losses.

4. SSL Evaluation

Three main way will be considered to evaluate the performances of a backbone model obtained from self-supervised learning:

- **Linear Probe**: a linear layer trained on top of the frozen representations generated by a pretrained backbone. This method evaluates how well a simple linear classifier can predict labels using these representations as input.

- **MLP**: same idea as Linear Probe, but this method considers a MLP instead of a single linear layer.

- **k-NN Classifier**: after freezing the pretrained backbone to compute and store the features of the training data of the downstream task, the nearest neighbor classifier then matches the feature of a test image to the k nearest stored features that votes for the label.

All of these methods implementations are in the github link in Sec. 2. Initially, a pretrained ResNet18 was considered as the backbone the first two evaluations have been applied for 10 epochs, while all the three methods evaluates the learned representations with Cifar10 dataset.
The results are in Tab. 1, by looking at them they seem all comparable, the best accuracy is reached with linear probe and in general the classifier (linear probe or MLP) learnt on top of the backbone network seems slightly better.
In Fig. 5 there is a brief analysis of the optimal value of k (number of nearest neighbors considered). Within the range considered for the value of k (5-200), the results do not vary too much, but the highest accuracy score is achieved with the values 20 and 40, although 20 is probably a more stable value since the scores associated with values near it are higher than the ones with 40.

Eval approach	Test Accuracy
Linear Probe	**0.41**
MLP with 2 layers	0.40
k-NN classifier	0.37

Table 1. Table with classification accuracy values varying with evaluation approach. All of them uses as backbone model the pretrained ResNet18 with ImageNet from torchvision.

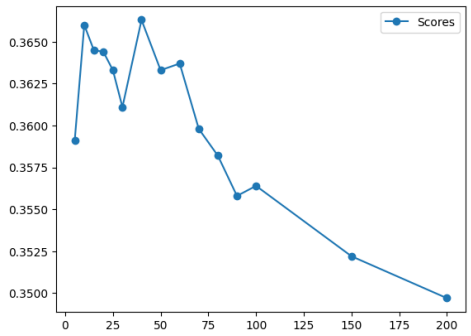

Figure 5. Accuracy for the ResNet18 pretrained backbone from scratch and the CIFAR10 dataset while changing k in the k-NN classifier.

After this initial attempt, a ResNet18 trained from scratch with supervised learning and Cifar10 dataset and evaluated with SSL methods was compared with the performances of the supervised models contained in the first report(https://github.com/Sossio699/ComputerVision/blob/main/Report%201%20CNN/Arati_report1_CNN.pdf). The results are contained in Tab. 2.
Linear probe remains the best technique, its results are even better than fine-tuning, while k-NN is the worst with this setup, and by considering 5000 nearest neighbor the computational time is also the highest of the considered approaches.

Training approach	Test Accuracy
Best from Scratch	0.67
Best with fine-tuning	0.76
Eval with Linear Probe	**0.82**
Eval with MLP 2 layers	0.81
Eval with 5000-NN classifier	0.62

Table 2. Table with classification accuracy values varying with training and evaluation approach. The first row refers to a custom CNN model, while the other rows use a ResNet18 model from torchvision. Linear Probe and MLP were applied for 10 epochs.

The fact that it requires 5000 nearest neighbors to achieve acceptable results, leaving aside the applied supervised training, is probably due to the training on CIFAR-10, which, having only 10 classes, might lead to representations that are easier to find but may not achieve good class separation.
Lastly, three models from timm package [4] were considered, they are pretrained with ImageNet-1k dataset and they could be evaluated both as image classifiers or feature backbones (by removing the classification head).
These three models are indicated in Tab. 3, and the table also contains the best classification results achieved with k-NN classifier varying the value of k (like what was done in Fig. 5).

timm Model	Test Accuracy
resnetv2_18ra4_e3600_r224_in1k	0.19
vit_small_patch8_224dino	**0.93**
convnextv2_nanofcmae	0.27

Table 3. Table with classification accuracy values varying with models from timm package. Each model is evaluated again with Cifar10 dataset and with the k-NN classifier technique.

As showed in the results, the second model, a Visual Transformer (ViT) pretrained with DINO technique (SSL approach based on Knowledge Distillation), completely outperforms the other two models (a ResNet and a CNN).
This outcome was predictable, since ViTs are known to exchange high computation requirement with an excellent ability to extract a global representation for large datasets, and this is confirmed even if the evaluation dataset is different from the one used to pretrain the model.
This result should ensure a strong ability to generalize to other downstream tasks.

5. Conclusions

This report has explored the fundamentals of Self-Supervised Learning and its ability to exploit large unlabeled data for representation learning.
Three main methods, based on the Siamese Network model architecture, were presented and analyzed: SimCLR, BYOL and Barlow Twins.
SimCLR relies on contrastive learning, requiring negative samples to avoid representation collapse, but produces strong results with sufficient computational resources.
BYOL eliminates the need for negative samples through an online-target network architecture with momentum updates, simplifying the training process while achieving competitive performance.
Barlow Twins takes a redundancy reduction approach, encouraging alignment of embeddings while avoiding over-correlation, making it computationally efficient and robust.
Also three methods to evaluate representations learned by SSL have been presented: Linear Probe, MLP and k-NN classifier. All of these techniques were compared to some of the classification accuracy results obtained with supervised learning, and in general they performed slightly better, considering that the backbone model were trained with a labeled dataset.
The k-nn Classifier technique was also applied to three pretrained models from the timm package, showing the different outcomes varying with the backbone architecture and the pretraining method. Visual Transformer stands out as the best model to learn global representations, although it requires lots of computational resources.
After having presented and analyzed some SSL training and evaluation methods, this final part aimed to provide a brief indication of the best architectures to use for representation learning (even though the considered models are pretrained with different techniques), especially with large datasets which could indeed represent a plausible scenario in the context of SSL.

References

[1] Mohammad Norouzi 01Ting Chen, Simon Kornblith and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. 2020. 2

[2] Florent Altche Corentin Tallec Pierre H. Richemond Elena Buchatskaya Carl Doersch Bernardo Avila Pires Zhaohan Daniel Guo Mohammad Gheshlaghi Azar Bilal Piot Koray Kavukcuoglu Remi Munos 02Jean Bastien Grill, Florian Strub and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. 2020. 2

[3] Ishan Misra Yann LeCun 03Jure Zbontar, Li Jing and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. 2021. 3

[4] Ross Wightman. timm - pytorch image models. https://huggingface.co/docs/timm/index. 5